Microsoft Visual C++／Visual Basic

Win32 API

オフィシャル リファレンス

マイクロソフト株式会社 監修

アスキー書籍編集部 編

主要
300API

アスキー出版局

Microsoft Visual C++／Visual Basic

Win32 API

オフィシャル リファレンス

主要
300API

マイクロソフト株式会社 監修

アスキー書籍編集部 編

アスキー出版局

表紙写真提供：IPS

はじめに

1995年11月にWindows 95が発売され、もはやほとんどパーソナルコンピュータはWindows 95をプラットフォームにし、より安全でより高速、そしてより簡単なアプリケーションにシフトしつつあります。これらのアプリケーションは、Windows 95やWindowsNT 3.51に搭載されているWin32 APIと呼ばれる32ビット対応のAPIをベースにビルドされています。APIとは、MS-DOSで言うところのファンクションコール（int 21h）に相当するものです。つまり、アプリケーションはおろかWindowsのシステム自体がこのWin32 APIの上に成り立っているのです。

しかし、Win32 APIに関するドキュメントは、すべて英語で提供されており、その量も膨大です。そこで、本書ではWindowsプログラミングをする上で必須となるAPIを300個に絞り込み説明しています。Visual C++からはもちろん、簡単にAPIを呼び出せるVisual BasicからのAPIの呼び出し方についても併記しました。

本書では、Visual C++やVisual Basicでの言語形態や開発環境の操作は説明していません。これらの情報は、添付のマニュアルや市販書籍をご覧ください。また、Windowsプログラミングの基礎について、第1章で簡単に紹介はしていますが、これだけではすべてを語れません。これらの情報については、市販書籍などを合わせてご覧ください。何かの参考になるかと思い、次に参考資料も掲載します。

また、本書の内容については万全を期していますが、プログラム開発を行う場合は、各種ヘッダファイルやSDKなどと合わせて十分なチェックを行ってください。

●参考資料一覧

・アーキテクチャや概論関連

『Windows 95 プレリリースガイド』　株式会社アスキー刊　2,800円

Windows 95のアーキテクチャの概論を簡単にまとめた書籍。

『INSIDE WINDOWS 95』　株式会社アスキー刊　4,800円

上記の『Windows 95 プレリリースガイド』をさらに掘り下げた書籍。

・プログラミング関連

『Windows 95 プログラミング 1』　株式会社アスキー刊　3,800円

Windowsの仕組みからプログラミングの基礎まで扱った心者向け書籍。

『Microsoft Windows 95 プログラマーズガイド』

株式会社アスキー刊　11,500円

コードを中心にしたWin32 APIに関したハイエンド開発者向け書籍。

『Windows 95 APIケーススタディ』　株式会社翔泳社刊　5,800円

サンプルプログラムを中心にした解説。VC++ 4.0用API学習書籍。

・オンライン関連

『Microsoft for Developers Only』　http://microsoft.com/devonly/

最新SDKなどを入手可能な開発者用情報サイト。

運営：Microsoft Corporation.

『The Development Exchange』　http://www.windx.com/

VB、VC++など豊富な技術情報が揃う。メール配信サービスは無料登録。

運営：Fawcette Technical Publications

・ソフトウェア

『Microsoft Visual Basic Version 4.0』

Standard Edition	30,000円	
Professional Edition	78,000円	
Enterprise Edition	128,000円	
	マイクロソフト株式会社	

『Microsoft Visual C++ Version 4.0』

デベロップメントシステム	98,000円	マイクロソフト株式会社

『Window SDK』

MSDN LEVEL2（年間契約）	76,220円	マイクロソフト株式会社

※価格はすべて1996年5月現在調べ。

目　次

第 1 章

WindowsアプリケーションとWin32 API

1.1 Windowsアプリケーションの動作

ここでは、WindowsシステムとWindowsアプリケーションがどのように連携して動作しているかを説明する。

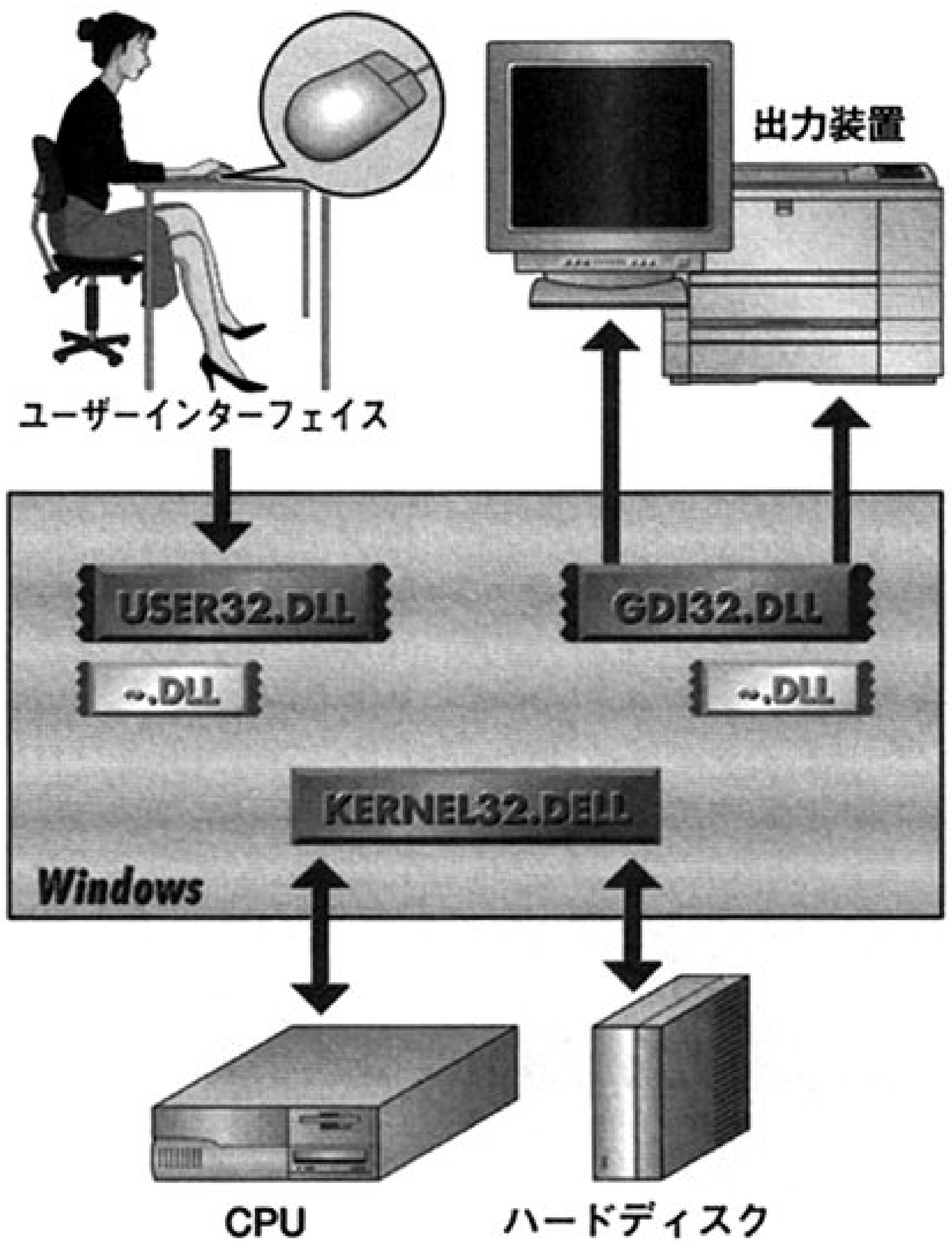
1.1.1 Windowsシステムの構成

Windows Operating Systemは、単体のプログラムモジュールで構成されたものではない。DLL（ダイナミックリンクライブラリ）と呼ばれる多数のライブラリを利用してつくられた一連のシステムがWindowsである。

DLLの中でも非常に重要な役割を担うのが"kernel32.dll"，"gdi32.dll"，"user32.dll"である。これらのDLLは、Windowsシステム自体を構成するだけではなく、アプリケーションを開発する際に避けては通れない関数を多数実装している。

それぞれのDLLは役割分担されており、DLLごとに管理するハードウェアやソフトウェアのリソース（資源）が異なる。

"kernel32.dll"には、メモリや外部記憶装置などのハードウェアを管理するための関数がまとめられている。"gdi32.dll"には、線を描いたり色を塗ったりといったグラフィックの描画を中心とした関数がまとめられている。また"user32.dll"には、ウィンドウやボタン、メニュー操作などのユーザーインターフェイスを管理するための関数がまとめられている。



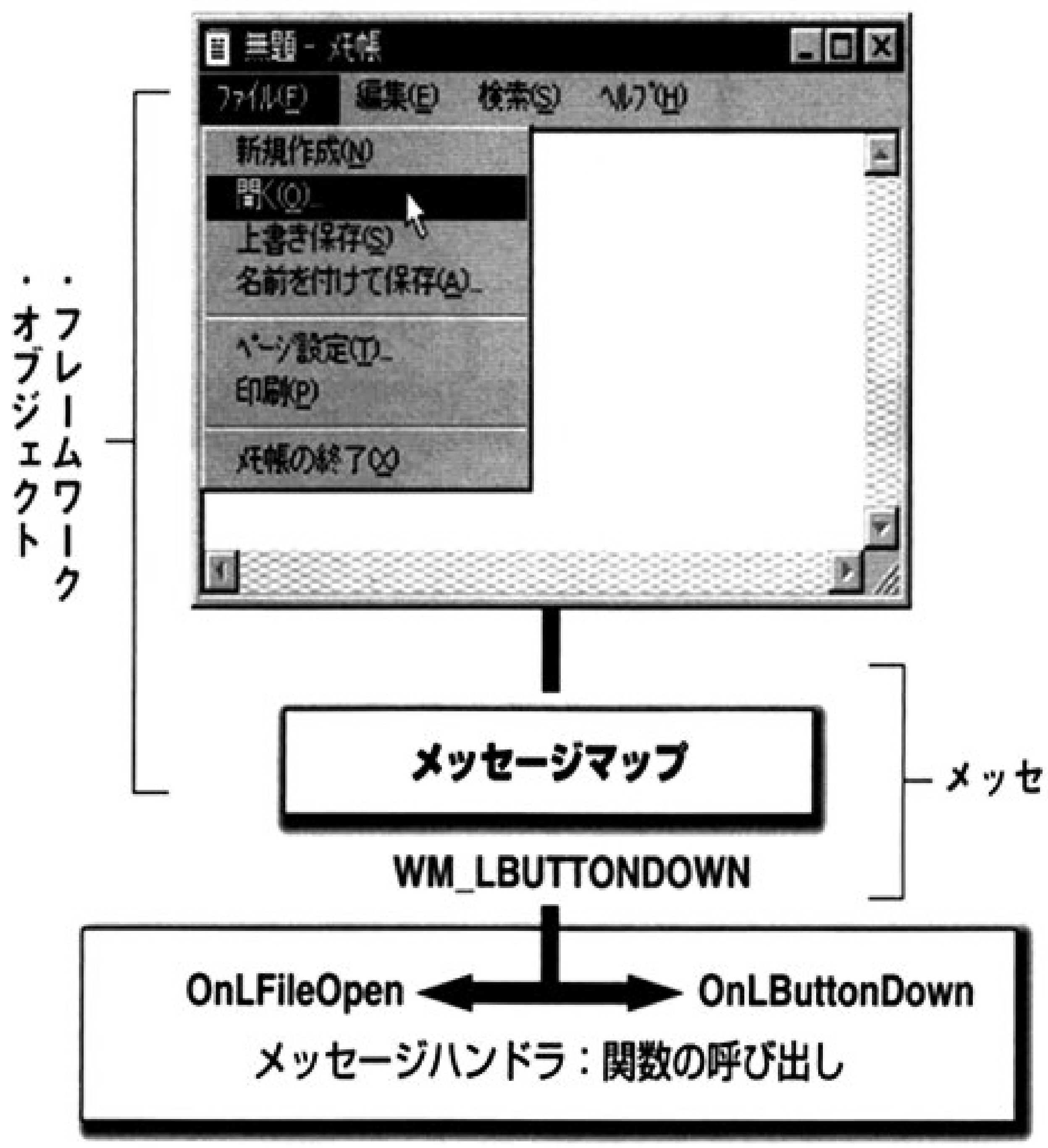
1.1.2 Windowsアプリケーションの特徴

Windowsアプリケーションの最大の特徴は、ユーザーがメニューを選択したり、タイマーが指定時間に達したりといったアクション（イベント）をきっかけにプログラムが実行されることである。

なんらかのイベントが発生するとWindowsのシステムがこれを感知し、これを「メッセージ」と呼ばれる形に変換する。このメッセージは、Windowsで実行中のアプリケーションに送信される。メッセージを受け取ったアプリケーションは、ここではじめてメッセージに応じた処理を行うのである。

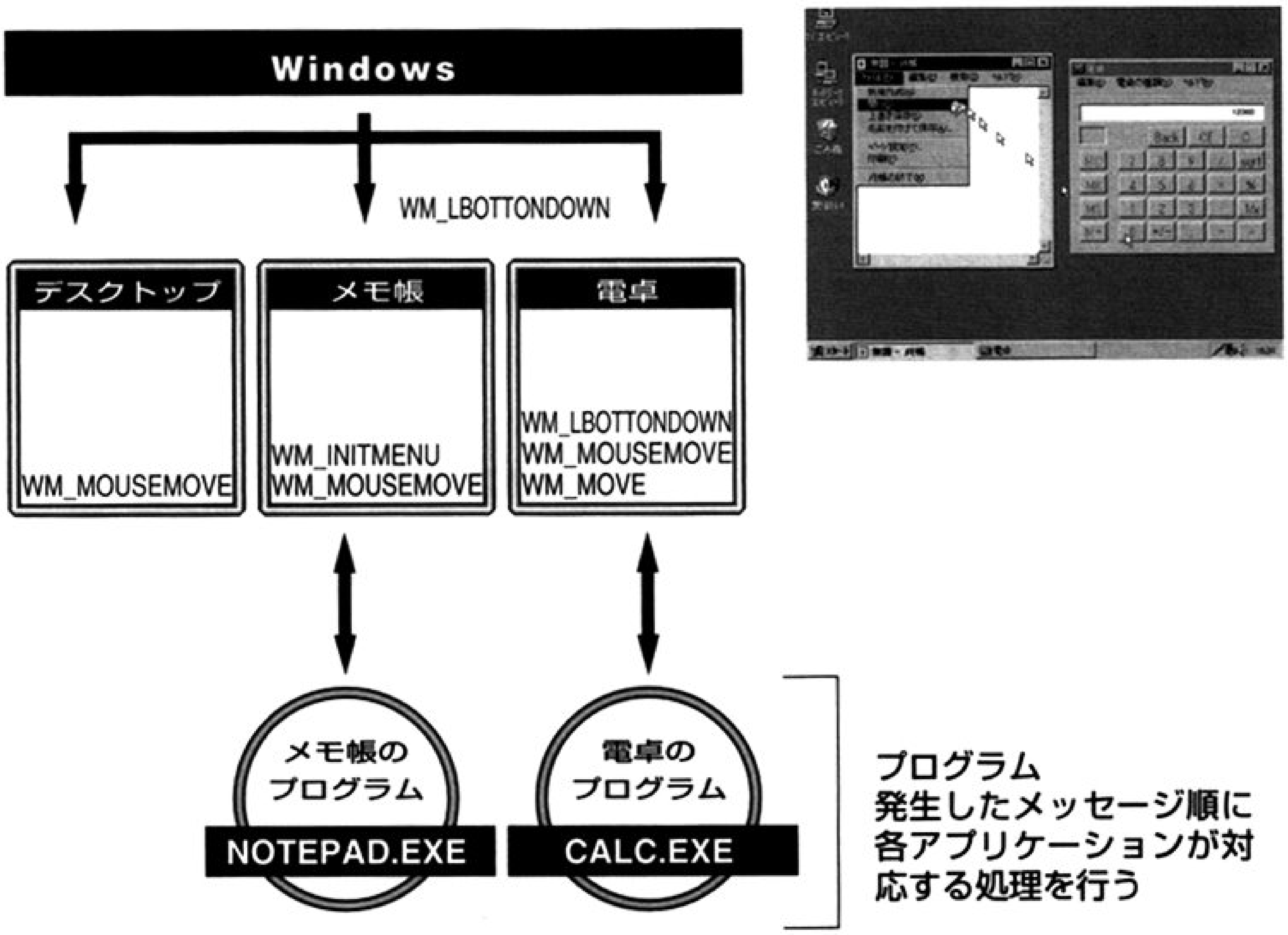
このようにWindowsアプリケーションは、メッセージ（イベント）により駆動するためイベント駆動型のアプリケーションと呼ばれる。

メッセージ駆動型プログラム



1.1.3 メッセージ

Windowsシステムでは、イベントに対応して多数のメッセージが発生している。このメッセージをキューに貯めて、各アプリケーションに振り分ける。



Windowsのシステムで発生する主要なメッセージには、WMで始まる名前が付けられている。主要なWindowsメッセージは、以下の通りである。

主要メッセージの一覧表	
WM_COMMAND	コマンドメッセージが送られてきた
WM_CREATE	ウィンドウの作成の指示
WM_DESTROY	ウィンドウの破棄の指示
WM_DROPFILES	ファイルがドロップされた
WM_HSCROLL	水平スクロールバーがクリックされた
WM_INITDIALOG	ダイアログボックスの初期化要求
WM_INITMENU	メニューがアクティブになる通知
WM_LBUTTONDBLCLK	マウスの左ボタンがダブルクリックされた
WM_LBUTTONDOWN	マウスの左ボタンが押された
WM_LBUTTONUP	マウスの左ボタンが離された
WM_MOUSEMOVE	マウスカーソルが移動した
WM_MOVE	ウィンドウの位置が変更された
WM_QUIT	アプリケーションの終了が要求された
WM_PAINT	ウィンドウの再描画の要求
WM_RBUTTONDBLCLK	マウスの右ボタンがダブルクリックされた
WM_RBUTTONDOWN	マウスの右ボタンが押された
WM_RBUTTONUP	マウスの右ボタンが離された
WM_SYSTEMERROR	システムエラーが発生した
WM_TIMER	タイマのタイムアウト時間が経過した
WM_VSCROLL	垂直スクロールバーがクリックされた

1.1.4 リソース

Windowsプログラムでは、よく「リソース」という言葉が使われる。直訳すると「資源」となる。「リソースが足りない」という言葉をよく耳にするが、日本語では「資源が足りない」と訳せるだろう。つまり、プログラムで利用している何らかの資源が足りないということだ。では、リソースの実体を説明する。

●ハードウェアリソース

プログラムから利用できる一番身近な物は、メモリやハードディスクだろう。通信用ポートやプリンタ、CPUもそうだ。つまり、コンピュータを構成している部分は、すべてリソースと言える。

●ソフトウェアリソース

プログラムから利用できるハードウェア以外の物と言えば、ビットマップやフォントがそうだ。次にWindowsプログラムでリソースと呼ばれるリストを掲載する。

主要なリソース		
アクセラレータ	ビットマップ	カーソル
ダイアログ	アイコン	メニュー
ストリングテーブル	ツールバー	バージョン

Windowsアプリケーションの実行ファイル（EXE）や後述のDLLファイルには、リストのリソースを組み込むことが可能である。それ以外にも、バイナリデータを含むことができる。

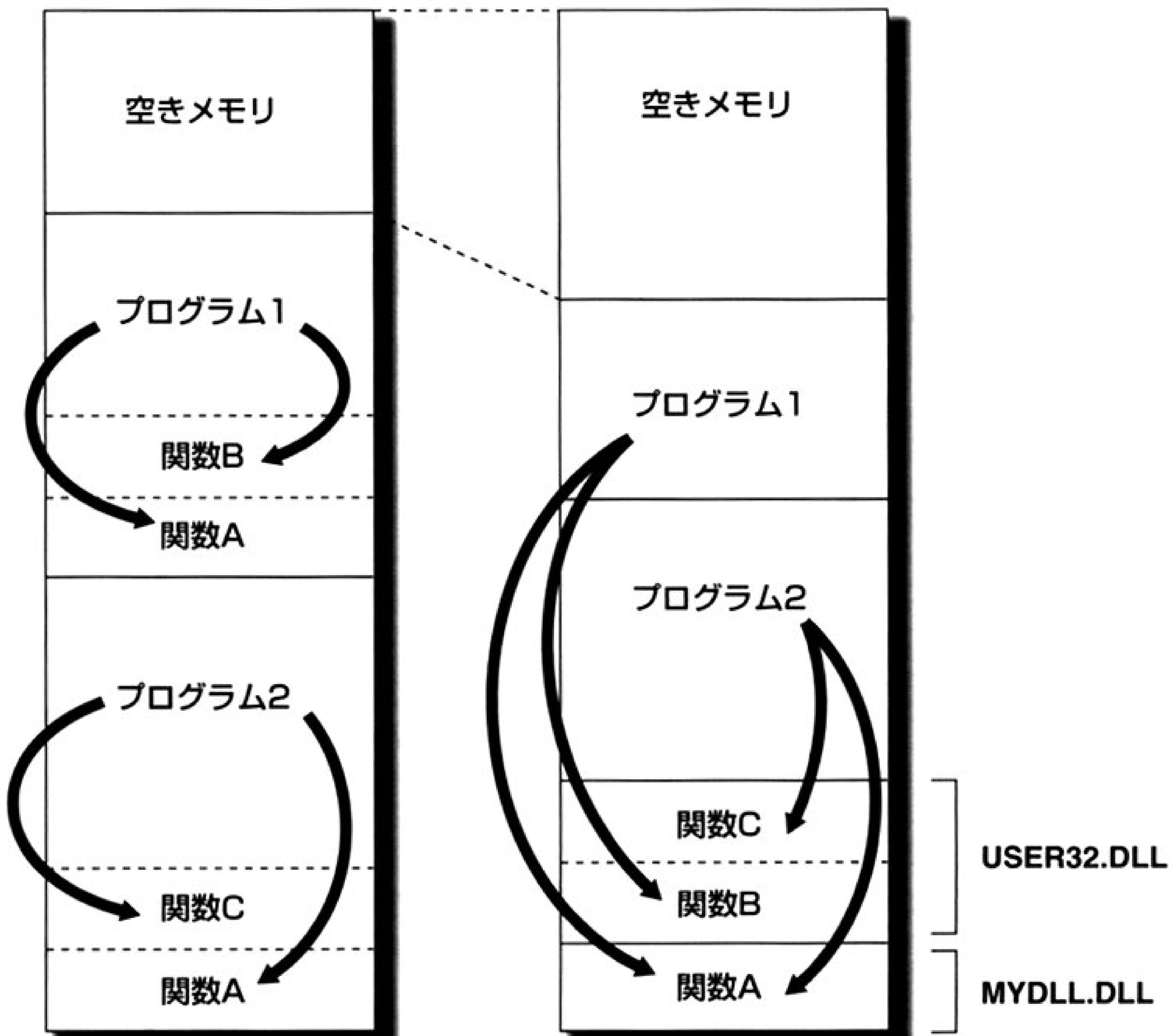
作成したリソースはリソースコンパイラを使用してコンパイルし、プログラムのオブジェクトファイルとリンクして実行ファイルに格納する。

プログラム中では、LoadResource,LoadMenuやLoadIconなどのAPI関数で呼び出すことができる。プログラム中で個々のリソースを指定できるように、リソース名やIDを割り当てておくことが必要である。

1.1.5 DLL - ダイナミックリンクライブラリ -

MS-DOSアプリケーションのように、利用する関数をすべて実行ファイルに組み込むことは可能である。しかし、複数のアプリケーションが動作するWindowsでは、同じような関数がアプリケーションごとに組み込まれ、メモリの無駄遣いになってしまう。

可能な限りリソース（資源）の共有をはかり、ハードディスクなどの記憶装置を始めメモリリソースなどの消費を最小限に抑えるアプリケーションを開発するにこしたことはない。



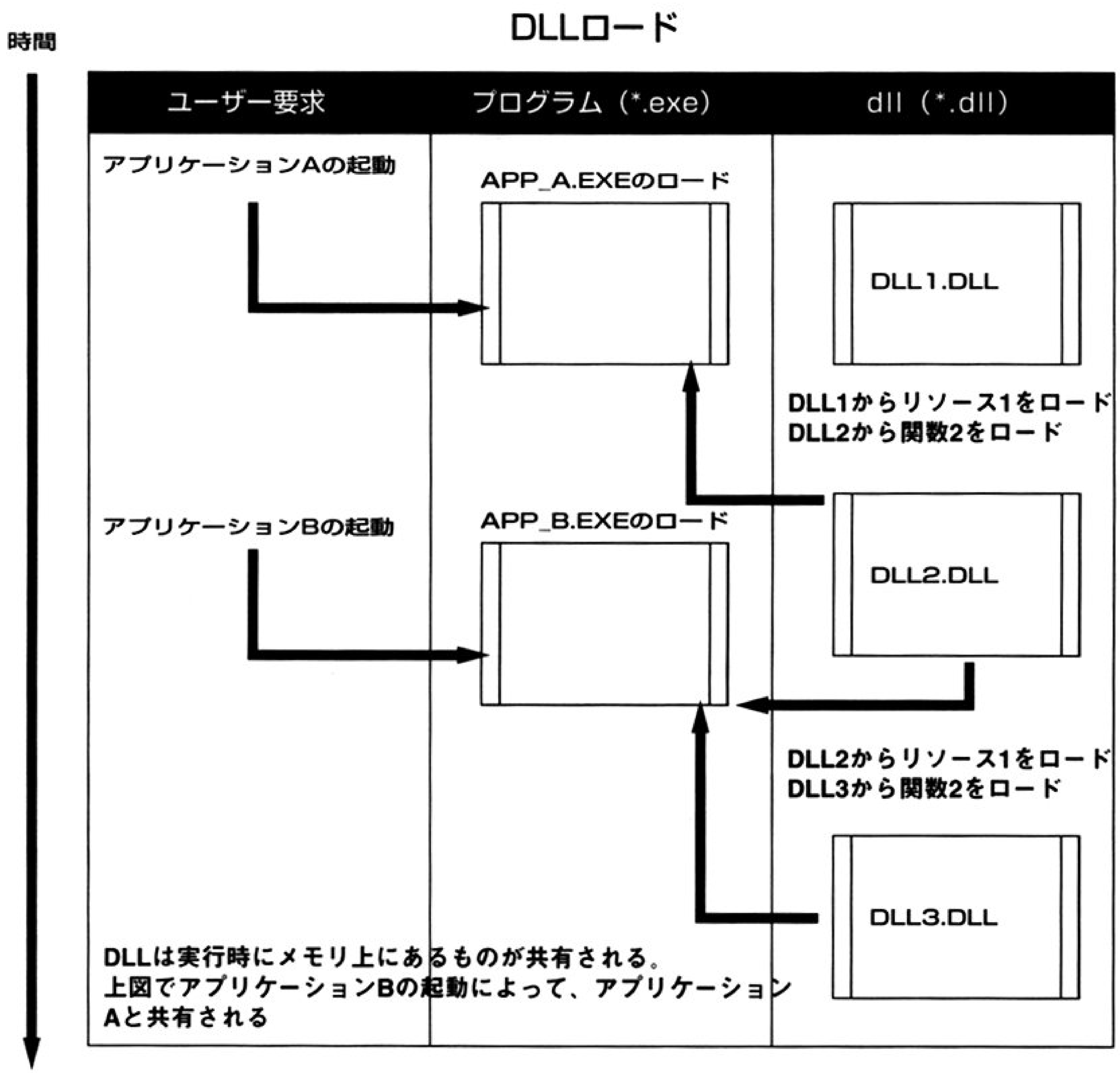
使用する関数をすべて実行プログラムに組み込んだ場合

使用する関数を動的に組み込み複数のプログラムで共有する場合

そのためWindowsでは、ライブラリの関数をアプリケーションから動的に呼び出す機能が提供されている。この機能を利用すると、呼び出そうとしている関数が、すでに別のアプリケーションによりメモリに常駐されていれば、その関数を共有することが可能である。これにより、メモリをより有効に利用できる。

このように動的に呼び出される関数群を実装したファイルは、DLL（Dynamic Link Library：ダイナミック・リンク・ライブラリ）と呼ばれる。DLLからの関数の読み込みは、実行アプリケーションの起動時もしくは、プログラム中の任意の箇所から行える。

DLLから読み込まれた関数はメモリに常駐し、他のアプリケーションから読み込みが要求された場合に共有される仕組みになっている。DLLには関数ばかりではなく、アイコンやメニュー、マウスカーソルなどの各種のリソースも実装可能である。



1.1.6 MS-DOSプログラムからの移行

MS-DOSはシングルタスクのOSだったため、MS-DOSアプリケーションがリソースを独占してもなんら問題はなかった。しかし、WindowsはマルチタスクOSのため、Windowsアプリケーションは、システムのさまざまなリソースを共有できるようにしなければならない。

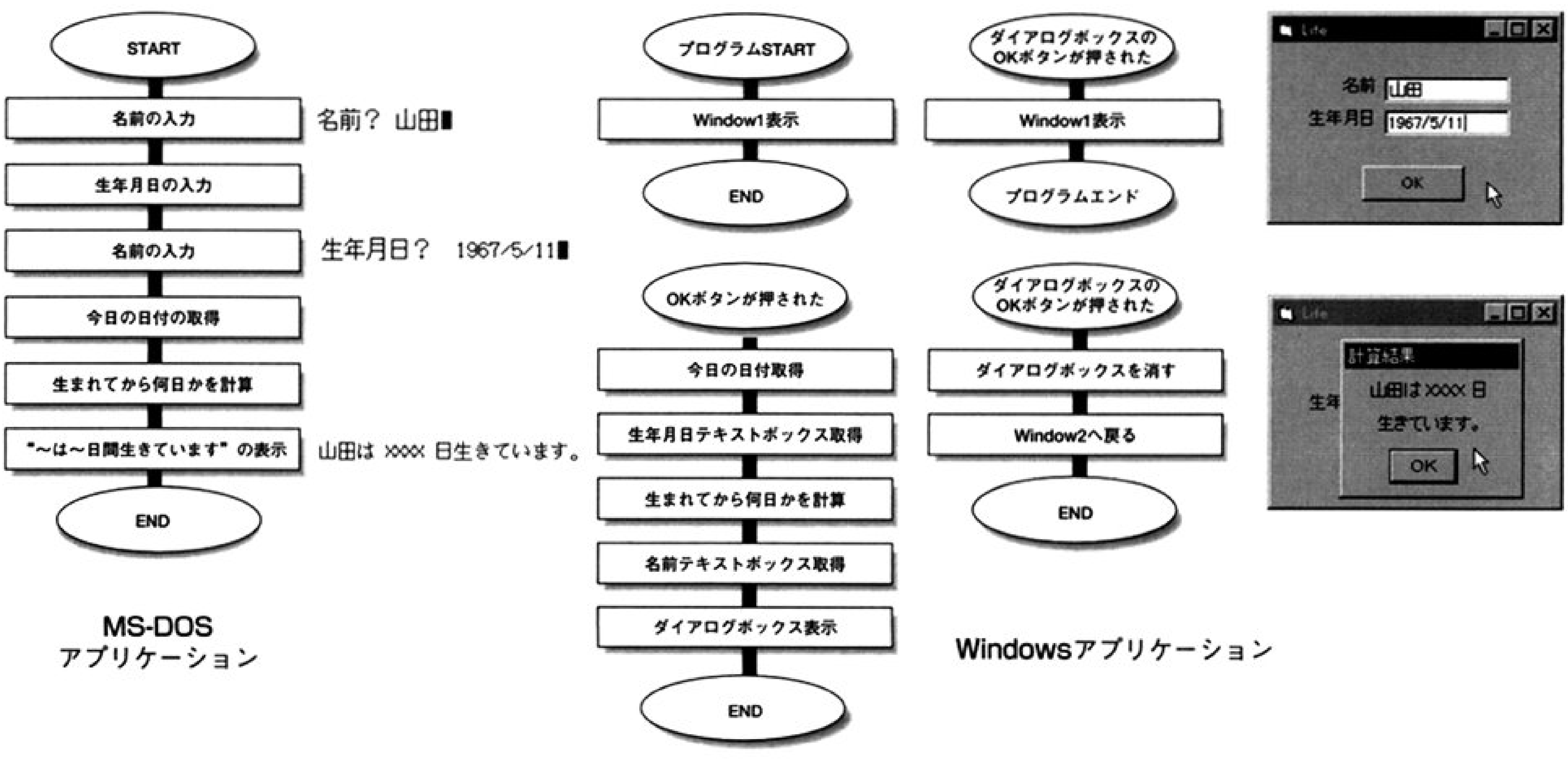
またWindowsアプリケーションは、イベント駆動型のため従来のMS-DOSアプリケーションのように、一連の手続きをコーディングしていくプログラミングスタイルとは異なっている。

ここでは、今までMS-DOSアプリケーションを開発していた人たちが、Windowsアプリケーションを開発する際に注意しなければならないことを説明する。

●プログラミングスタイルの違い

MS-DOSアプリケーションはCPUの制御を行うことができ、ハードウェアやソフトウェアの資源を独占することが可能だった。これは言い換えれば、すべての管理をアプリケーションで処理しなくてはならず、ユーザーに対するインターフェイスなどの入出力部分もアプリケーションごとに作成しなくてはならないということである。

これに対してWindowsアプリケーションは、前述の通りイベント駆動アプリケーションである。たとえば、実行中のアプリケーションのウィンドウの中でマウスをクリックすると（マウスクリックのイベントが発生すると）、Windowsのシステムがアプリケーションに対して、マウスがクリックされたというメッセージを送る。アプリケーションプログラム側では、メッセージに応じた処理を行う。



また、Windowsではハードウェアの違いをWindowsのシステム側で吸収し、アプリケーション側ではほとんど気にする必要はない。たとえば、様々な種類のビデオアクセラレータボードが存在するが、適合したディスプレイドライバを使用すれば、アプリケーション側では解像度や色数のみに留意すれば良い。

●GUI

MS-DOSアプリケーションとWindowsアプリケーションを比較すると、一番大きな違いはGUI（グラフィックユーザーインターフェイス）であろう。Windowsアプリケーションは名前の通り、ウィンドウシステムのGUIを採用している。

これに対してMS-DOSはCUI（キャラクタユーザーインターフェイス）である。ユーザーインターフェイスの基本は文字ベースである。もちろんGUIを採用したアプリケーションも存在するが、アプリケーション独自のGUIのサブシステムを組み込んでいるためである。WindowsはOSがGUIベースで設計されているので、容易にGUIウィンドウを採用したプログラムが作成できる。

さらに、アプリケーションが異なっても同様のインターフェイスで操作できるので、アプリケーションごとに操作を覚える必要が少なくなる。

●マルチタスク環境

Windowsはマルチタスクを実現するため、ハードウェア・ソフトウェアのリソースを複数のアプリケーションが共有できるようになっている。しかし、かつての16ビットWindowsではCPUが直接タスクを切り替えるのではなくアプリケーションに動作を任せた、疑似マルチタスクであるため、1つのアプリケーションがタスクを独占してしまうと、他のアプリケーションに制御が移らない場合もあり得た。

現在主流の32ビットWindows（Windows 95、Windows NT）ではプリエンプティブなタイムスライス・マルチタスキングがOSによって実行されているので、基本的にあるアプリケーションプログラムがシステムを占有してしまうことはできない。

●API

Windowsには多数のWindows API（アプリケーションプログラムインターフェイス）関数が用意されている。このAPI関数とはWindowsから提供されるMS-DOSのファンクションコールのようなものである。MS-DOSプログラムではBIOSを使用しなくてもアプリケーションは作成できるが、WindowsプログラムはAPIを使用しなくては何もできない。ウィンドウの表示から文字の出力まで、すべてAPIを呼び出す必要がある。もちろん、直接ファンクションコール（int 21h）を呼び出すこともできない。

●ファイル形式

Windowsの実行ファイル形式の拡張子はEXEであるが、内部はMS-DOSのファイルの構造とは異なっている。MS-DOSで作成した拡張子がEXEまたはCOM形式のファイルを実行すると、MS-DOSモードで実行される。WindowsプログラムのEXEファイルには、アイコンやメニューなどのリソースを含むことができる。

また、Windowsでは拡張子がDLLのファイルを動的にロードすることができる。DLLファイルは、WindowsシステムのAPIと同様に呼び出しが可能である。DLLファイルは関数を含んだファイルと考えると良いだろう。もちろんリソースも含むことができる。

1.2 Win32 API

WindowsアプリケーションとWindows APIが切っても切り離せない関係にあるのは、十分理解していただけたであろう。CPUが8ビット、16ビット、32ビットと進化すると同時に、OSも進化を遂げている。Windowsもまたしかり、Windows 3.1では「Win16 API」と呼ばれる16ビットWindows APIが実装されていたが、WindowsNT 3.51やWindows 95では32ビットAPIの「Win32 API」が実装されている。

ここでは、Win32 APIを利用する上での注意点などを説明する。

1.2.1 Win32 APIの機能概要

32ビット対応のWindowsアプリケーションを実行させるには、Windows 3.1にWin32sと呼ばれる追加APIセットを導入する方法と、Windows 95やWindows NTを利用する方法がある。Windows 3.1＋Win32sは、サブセットAPIのため本書では触れない。

次表は、Win16 APIからWin32 APIへ移行する中で実現されてきた機能比較である。表ではAPI別ではなく、プラットフォームとなるOS別に機能を比較している。

機能	3.1	95	NT
32ビット化	×	○	○
自動ハードウェア認識	×	○	○
プラグアンドプレイ	×	○	×
プリエンプティブ・マルチタスク	×	○	○
Win 32 API	×	○	○
Win 32s API	○	○	○
ネットワーク機能	×	○	○
16ビットアプリケーション間のクラッシュ保護	×	×	○
セキュリティ	×	×	○
システム自動復帰	×	×	○
MS-DOSサポート	○	○	△
コンソールAPI	×	×	○
OpenGLサポート	×	○	○
MS-DOSリアルモードドライバ利用	○	○	×
16ビットWindowsドライバ利用	○	○	△
UNICODE対応	×	○	○
ディスク圧縮機能	○	○	○
非Intel CPUサポート	×	×	○
マルチプロセッサ・サポート	×	×	○

注：WindowsNTはVersion 3.51

1.2.2 Win16からの移行

Win16 APIとWin32 APIは、ほとんどのAPIで互換性が保たれている。既存の16ビットアプリケーション資産（Visual C++ 1.0、Visual C++ 1.5、Visual Basic 2.0など）はソースコードの改変をほとんど必要とせず、32ビットアプリケーション化できる。ただし、一部のメッセージや、関数の戻り値など細かな変更がされている部分が少なくない。このあたりには十分な注意が必要になる。詳細についてはWin 32 SDKを参照されたい。もう1点、32ビットアプリケーションを開発するためには当然32ビットコードを生成する開発環境が必要になる。これらがVisual C++ 4.xもしくはVisual Basic 4.xということになる。

32ビットプログラミングで開発者（アプリケーション利用者も）が受ける最も大きな恩恵として、システムリソースや64KBセグメントの撤廃がある。

次の表はWin32 APIで廃止もしくは拡張などの変更がなされたAPIおよびメッセージの一覧である。

サポート	API／メッセージ	備考
拡張	WndProc	ウィンドウプロシージャWndProc (HWNDhwnd,UINTmsg,UINTwParam,UINTlParam) を定義
拡張	wndproc	ウィンドウプロシージャWndProc (HWNDhwnd,UINTmsg,UINTwParam,UINTlParam) を定義
拡張	AddFontResource	ファイル名には、ハンドルではなく文字列を使用
拡張	EM_GETSEL	wParam/lParamパッキング変更
拡張	EM_LINESCROLL	wParam/lParamパッキング変更
拡張	EM_SETSEL	wParam/lParamパッキング変更
拡張	GetClassWord	Win32で32ビットに拡張されている値にはGetClassLongを使用
拡張	GetModuleUsage	Win32では常に1を戻す
拡張	GetWindowWord	Win32で32ビットに拡張されている値にはGetWindowLongを使用
拡張	RemoveFontResource	ファイル名にはハンドルでなく文字列を使用
拡張	SetClassWord	Win32で32ビットに拡張されている値にはSetClassLongを使用
拡張	SetWindowWord	Win32で32ビットに拡張されている値にはSetWindowLongを使用
拡張	WM_ACTIVATE	wParam/lParamパッキング変更
拡張	WM_CHANGECBCHAIN	wParam/lParamパッキング変更
拡張	WM_CHARTOITEM	wParam/lParamパッキング変更
拡張	WM_COMMAND	wParam/lParamパッキング変更
拡張	WM_DDE_ACK	wParam/lParamパッキング変更
拡張	WM_DDE_ADVISE	wParam/lParamパッキング変更
拡張	WM_DDE_DATA	wParam/lParamパッキング変更
拡張	WM_DDE_EXECUTE	wParam/lParamパッキング変更
拡張	WM_DDE_POKE	wParam/lParamパッキング変更
拡張	WM_HSCROLL	wParam/lParamパッキング変更
拡張	WM_MDIACTIVATE	wParam/lParamパッキング変更
拡張	WM_MDISETMENU	wParam/lParamパッキング変更
拡張	WM_MENUCHAR	wParam/lParamパッキング変更
拡張	WM_MENUSELECT	wParam/lParamパッキング変更
拡張	WM_PARENTNOTIFY	wParam/lParamパッキング変更
拡張	WM_VKEYTOITEM	wParam/lParamパッキング変更

サポート	API／メッセージ	備考
拡張	WM_VSCROLL	wParam/lParamパッキング変更
置換	WM_CTLCOLOR	WM_CTLCOLORtypeメッセージで置換。wParam/lParamパッキング変更
廃止	AccessResource	Win32APIには相当するものなし（リソースAPIで対応予定）
廃止	AllocDSToCSAlias	Win32APIには相当するものなし
廃止	AllocResource	Win32APIには相当するものなし（リソースAPIで対応予定）
廃止	AllocSelector	Win32APIには相当するものなし
廃止	ChangeSelector	Win32APIには相当するものなし
廃止	CloseComm	CloseFileで置換
廃止	CloseSound	マルチメディアサウンドサポートで置換
廃止	CountVoiceNotes	マルチメディアサウンドサポートで置換
廃止	DeviceCapabilities	移植可能なDeviceCapabilitiesExで置換
廃止	DeviceMode	移植可能なDeviceModeExで置換
廃止	DlgDirSelect	移植可能なDlgDirSelectExで置換
廃止	DlgDirSelectComboBox	移植可能なDlgDirSelectComboBoxExで置換
廃止	DOS3Call	名前付きの移植可能なWin32APIで置換
廃止	ExtDeviceMode	移植可能なExtDeviceModeExで置換
廃止	FlushComm	PurgeCommで置換
廃止	FreeSelector	Win32APIには相当するものなし
廃止	GCW_CURSOR	Get|SetClassCursorで置換
廃止	GCW_HBRBACKGROUND	Get|SetClassBrBackgroundで置換
廃止	GCW_HICON	Get|SetClassIconで置換
廃止	GetAspectRatioFilter	移植可能なGetAspectRatioFilterExで置換
廃止	GetBitmapDimension	移植可能なGetBitmapDimensionExで置換
廃止	GetBrushOrg	移植可能なGetBrushOrgExで置換
廃止	GetCodeHandle	Win32APIには相当するものなし
廃止	GetCodeInfo	Win32APIには相当するものなし
廃止	GetCommError	GetCommStateで置換
廃止	GetCurrentPDB	Win32APIには相当するものなし
廃止	GetCurrentPosition	移植可能なGetCurrentPositionExで置換
廃止	GetEnvironment	Win32APIには相当するものなし
廃止	GetInstanceData	等価なものなし。IPCメカニズムでサポート
廃止	GetKBCodePage	Win32APIには相当するものなし

サポート	API／メッセージ	備考
廃止	GetMetaFileBits	移植可能なGetMetaFileBitsExで置換
廃止	GetTempDrive	Win32には適用されず
廃止	GetTextExtent	移植可能なGetTextExtentPointで置換
廃止	GetTextExtentEx	移植可能なGetTextExtentExPointで置換
廃止	GetThresholdEvent	マルチメディアサウンドサポートで置換
廃止	GetThresholdStatus	マルチメディアサウンドサポートで置換
廃止	GetViewportExt	移植可能なGetViewportExtExで置換
廃止	GetViewportOrg	移植可能なGetViewportOrgExで置換
廃止	GetWindowExt	移植可能なGetWindowExtExで置換
廃止	GetWindowOrg	移植可能なGetWindowOrgExで置換
廃止	GlobalDosAlloc	Win32APIには相当するものなし
廃止	GlobalDosFree	Win32APIには相当するものなし
廃止	GlobalPageLock	Win32APIには相当するものなし
廃止	GlobalPageUnlock	Win32APIには相当するものなし
廃止	GWW_HINSTANCE	GWL_HINSTANCEで置換
廃止	GWW_HWNDPARENT	GWL_HWNDPARENTで置換
廃止	GWW_ID	GWL_IDで置換
廃止	GWW_USERDATA	GWL_USERDATAで置換
廃止	LimitEMSPages	Win32APIには相当するものなし
廃止	LocalNotify	Win32には相当するものなし
廃止	MoveTo	移植可能なMoveToExで置換
廃止	NetBIOSCall	名前付きの移植可能なWin32APIで置換
廃止	OffsetViewportOrg	移植可能なOffsetViewportOrgExで置換
廃止	OffsetWindowOrg	移植可能なOffsetWindowOrgExで置換
廃止	OpenComm	OpenFileで置換
廃止	OpenSound	マルチメディアサウンドサポートで置換
廃止	ProfClear	Win32プロファイル文字列APIで置換
廃止	ProfFinish	Win32プロファイル文字列APIで置換
廃止	ProfFlush	Win32プロファイル文字列APIで置換
廃止	ProfInsChk	Win32プロファイル文字列APIで置換
廃止	ProfSampRate	Win32プロファイル文字列APIで置換
廃止	ProfSetup	Win32プロファイル文字列APIで置換

サポート	API／メッセージ	備考
廃止	ProfStart	Win32プロファイル文字列APIで置換
廃止	ProfStop	Win32プロファイル文字列APIで置換
廃止	ReadComm	ReadFileで置換
廃止	ScaleViewportExt	移植可能なScaleViewportExtExで置換
廃止	ScaleWindowExt	移植可能なScaleWindowExtExで置換
廃止	SetBitmapDimension	移植可能なSetBitmapDimensionExで置換
廃止	SetCommEventMask	SetCommMaskで置換
廃止	SetEnvironment	Win32APIには相当するものなし
廃止	SetMetaFileBits	移植可能なSetMetaFileBitsExで置換
廃止	SetResourceHandler	Win32には相当するものなし（リソースAPIで対応予定）
廃止	SetSoundNoise	マルチメディアサウンドサポートで置換
廃止	SetViewportExt	移植可能なSetViewportExtExで置換
廃止	SetViewportOrg	移植可能なSetViewportOrgExで置換
廃止	SetVoiceAccent	マルチメディアサウンドサポートで置換
廃止	SetVoiceEnvelope	マルチメディアサウンドサポートで置換
廃止	SetVoiceNote	マルチメディアサウンドサポートで置換
廃止	SetVoiceQueueSize	マルチメディアサウンドサポートで置換
廃止	SetVoiceSound	マルチメディアサウンドサポートで置換
廃止	SetVoiceThreshold	マルチメディアサウンドサポートで置換
廃止	SetWindowExt	移植可能なSetWindowExtExで置換
廃止	SetWindowOrg	移植可能なSetWindowOrgExで置換
廃止	StartSound	マルチメディアサウンドサポートで置換
廃止	StopSound	マルチメディアサウンドサポートで置換
廃止	SwitchStackBack	Win32APIには相当するものなし
廃止	SwitchStackTo	Win32APIには相当するものなし
廃止	SyncAllVoices	マルチメディアサウンドサポートで置換
廃止	UngetCommChar	Win32には相当するものなし
廃止	ValidateCodeSegments	Win32APIには相当するものなし
廃止	ValidateFreeSpaces	Win32APIには相当するものなし
廃止	WaitSoundState	マルチメディアサウンドサポートで置換
廃止	WriteComm	WriteFileで置換

1.3 C++からのWindows API呼び出し

Visual C++において、Win32 APIはMFC（Microsoft Foundation Class Library）に隠蔽され、ほとんどAPIを呼び出す必要はない。というよりは、APIの呼び出しは極力避けて、可能な限りMFCで開発を進めることを勧める。MFCは非常に完成度の高いクラスライブラリになっており、これを利用することで開発工数は以前に比べ少なく見積もることが可能になる。しかし、残念なことにMFCだけではどうにも処理できない機能を実装しなければならない場合も少なからずある。このような場合はAPIを直接呼び出さざるを得ない。

Visual C++でのAPI呼び出しは、いたって簡単である。必要なヘッダーファイルのインクルードと、ライブラリのインポートを定義するのみでAPIを直接呼び出せる。ヘッダーファイルとインポートすべきライブラリに関しては、Visual C++に付属するWin32 SDKのWin32 Referenceを参照すれば解決できる。

1.4 CからのWindows API呼び出し

C言語でプログラムを作成する場合、MS-DOSのプログラムではmain関数が最初に実行される。Windowsでは、WinMain関数が用意されており、Windowsのシステムから最初に呼び出される。

しかし、Main関数とWinMain関数では動作はまったく異なっており、通常、WinMainでは初期化処理とメッセージの交換を行う。実際の処理の記述は、RegisterClass API関数で登録したメッセージの受け取り先の関数（MainWndProc関数などと指定することが多い）に記述しなくてはならない。

メッセージの受け取り先の関数では、Windowsのシステムより送られてくるメッセージの内容を判断して、対応した処理を記述する。次のプログラムはスケルトン（何もしない）Windowsプログラムの記述例である。実行するとウィンドウが表示されるだけであるが、Windowsシステムと実行したプログラムの間では様々なメッセージが飛び交っている。

たとえば、ウィンドウを作成した時点でWM_CREATEというメッセージがMainWndProcに送られてくる。表示されたウィンドウ上でマウスを動かすだけでもWM_MOUSEMOVEというメッセージが送られてくるのである。

もちろん、すべてのメッセージに対して処理を記述する必要はない。必要なイベントに対応するメッセージに対してのみ処理を記述すればよいのである。プログラムを終了しウィンドウを破棄する時にのみ、必ずPostQuitMessage API関数を呼び出さなくてはならない。

Win32ではhPrevInstanceは常にNULLが返され、インスタンスは共有されずに多重起動する。WinMain関数の型は、Win16ではPASCAL型であったが、Win32ではWINAPI型に変更されている。

MainWndProcなどのコールバック関数は、将来の拡張の為にCALLBACKを使用することが推奨されるようになった。

```
#include <windows.h>

LRESULT CALLBACK MainWndProc (HWND, UINT, WPARAM, LPARAM)
                                              ; /* ウィンドウ関数 */

int WINAPI WinMain (hCurInst, hPrevInst, lpCmdLine, iCmdShow)
HANDLE hCurInst ;       /* 現在のインスタンスハンドル */
HANDLE hPrevInst ;      /* 直前のインスタンスハンドル */
LPSTR  lpCmdLine ;      /* コマンドラインのロングポインタ */
int    iCmdShow ;       /* ウィンドウの表示状態値 */
{
MSG msg ;               /* メッセージ構造体 */

   if  (!hPrevInst)                      /* hPrevInstは常にNULLが返る */
   if  (!InitApplication (hCurInst))     /* ウィンドウの登録 */
      return  (FALSE) ;                  /* 登録失敗ならば終了 */

   if  (!InitInstance (hCurInst, iCmdShow)) /* ウィンドウの作成／表示 */
      return  (FALSE) ;                     /* 作成失敗ならば終了 */

    /* メッセージ・ループ */

    while  (GetMessage (&msg,                /* メッセージの取得 */
              NULL,
              NULL,
              NULL))  {
   TranslateMessage (&msg)  ;             /* メッセージの変換 */
   DispatchMessage (&msg)  ;              /* メッセージのディスパッチ */
    }

   return  (msg.wParam) ;     /* PostQuitMessageで設定された値を返し終了 */
}
```

```
BOOL InitApplication (hCurInst)
HANDLE hCurInst ;      /*現在のインスタンス*/
{
WNDCLASS wc ;    /* ウィンドウクラス構造体 */

    /* ウィンドウクラス構造体の設定 */

    wc.style            = NULL ;                 /*クラススタイル*/
    wc.lpfnWndProc      = MainWndProc ;          /*ウィンドウ関数*/
    wc.cbClsExtra       = 0 ;                    /*追加クラスデータ*/
    wc.cbWndExtra       = 0 ;                    /*追加ウィンドウデータ*/
    wc.hInstance        = hCurInst ;             /*現在のインスタンス*/
    wc.hIcon            = LoadIcon  (hCurInst, IDI_ICON)  ;
                                              /*ウィンドウで使用するアイコン*/
    wc.hCursor              = LoadCursor (NULL, IDC_ARROW)  ;
                                              /*ウィンドウで使用するカーソル*/
    wc.hbrBackground = GetStockObject  (BLACK_BRUSH)  ;
                                                          /*背景となる色*/
    wc.lpszMenuName  =  NULL ;                   /*メニュー名*/
    wc.lpszClassName = "TDBWinWClass" ;          /*ウィンドウクラス名*/

    /* ウィンドウクラスの登録 */

    return  (RegisterClass (&wc))  ;
                                         *ウィンドウクラス登録の結果を返す*/
}

BOOL InitInstance (hCurInst, iCmdShow)
HANDLE hCurInst ;      /*現在のインスタンス*/
int    iCmdShow ;      /*ウィンドウの表示状態値*/
{
HWND hwnd ;            /*作成したメインウィンドウのハンドル*/
HDC hdc;

    hInst = hCurInst ;   /*現在のインスタンスの保存*/

    /* ウィンドウの作成 */
    hwnd = CreateWindow (
        "TDBWinWClass",        /*RegisterClassで登録したウィンドウクラス名*/
        "TDB Sample app",      /*タイトルバーに表示する文字列*/
        WS_OVERLAPPEDWINDOW,                  /*ウィンドウスタイル*/
        CW_USEDEFAULT , CW_USEDEFAULT,
        CW_USEDEFAULT , CW_USEDEFAULT,
```

```
            NULL,                         /*親，オーナーのウィンドウは無し*/
            NULL,                         /*メニューは無し*/
            hCurInst,                     /*現在のインスタンス*/
            NULL                          /*ウィンドウ作成時の情報不要*/
        ) ;

    DeleteDC ( hdc) ;

     if  (!hwnd)                           /*ウィンドウは作成されたか？*/
         return  (FALSE)  ;                /*作成失敗ならばFALSEを返す*/

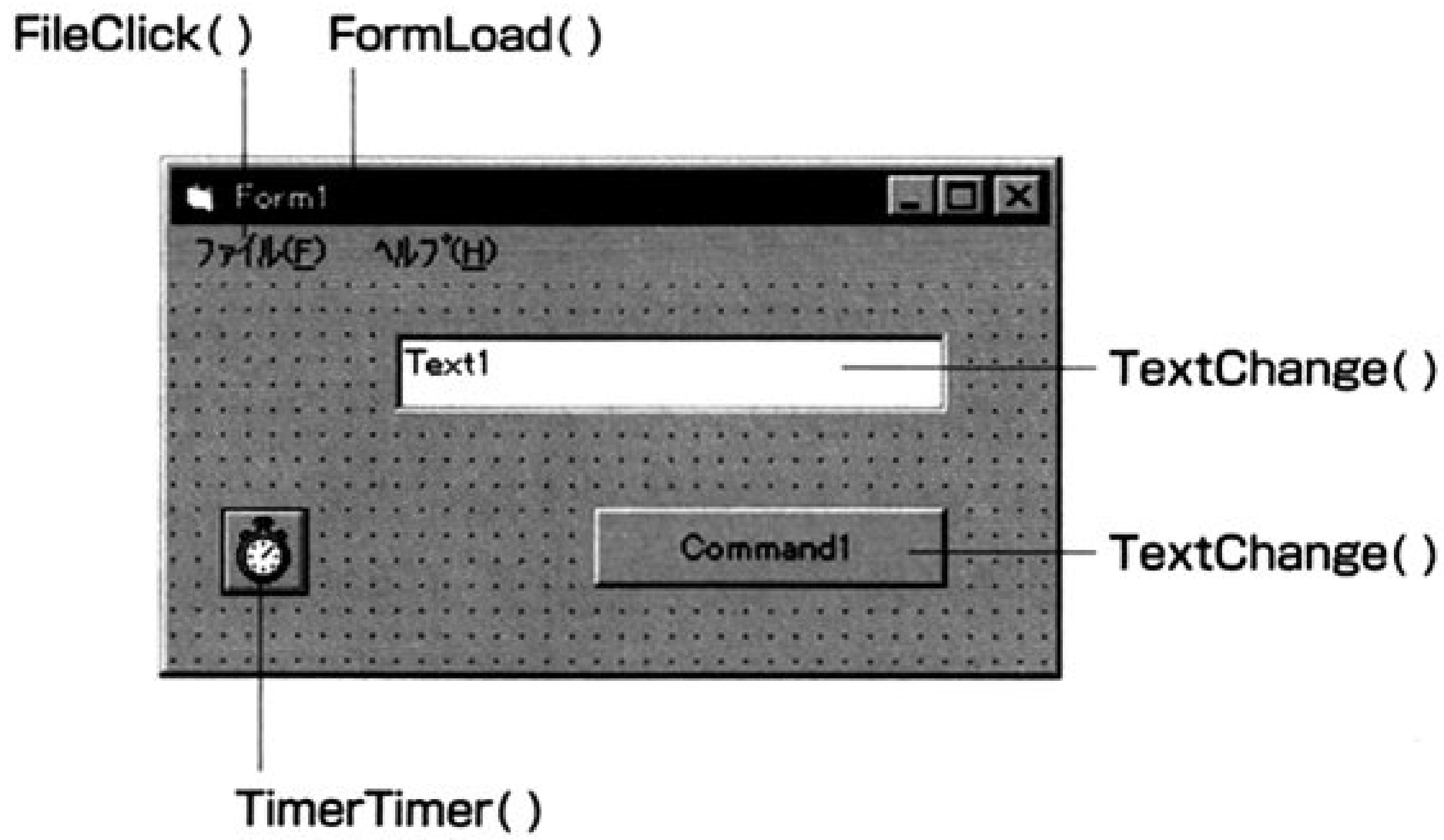
     ShowWindow (hwnd, iCmdShow)  ;
     UpdateWindow (hwnd)  ;
     return  (TRUE)  ;                     /*ウィンドウ作成成功を返す*/
}

LRESULT CALLBACK MainWndProc (hwnd, message, wParam, lParam)
HWND      hwnd ;          /*ウィンドウハンドル*/
UINT   message ;          /*メッセージ*/
WPARAM wParam ;           /*ワード値のメッセージ付加情報*/
LPARAM lParam ;           /*ロング値のメッセージ付加情報*/
{
     switch  (message)  {                  /*メッセージの判断 (32bit) */

   case WM_CREATE:
      break;
   case WM_DESTROY:                 /*ウィンドウ破棄のメッセージ*/
      PostQuitMessage (0)   ;    /*WinMan関数のメッセージループを抜ける処理
*/
      break;
   default:
       return  (DefWindowProc (hwnd, message, wParam, lParam))  ;
     }
     return  (DefWindowProc (hwnd, message, wParam, lParam))  ;
}
```

1.5 Visual BasicからのWindows API呼び出し

Visual Basicでは、主要なイベントそれぞれにサブルーチンが自動的に付加されるような仕組みとなっている。たとえば、マウスでボタンをクリックした時に、何かをさせたいと思ったら、フォーム上にボタンを配置してクリックするだけで対応するサブルーチンが現れる。ここにVisual Basicのコードを記述すれば良い。



Visual BasicにおいてAPIの呼び出しを行うには、3段階の手続きが必要である。

1. DeclareによるAPIの使用を宣言
2. 構造体の定義
3. APIの呼び出し

1.5.1 DeclareによるAPIの使用を宣言

SUBプロシージャでAPIを利用する前に、DeclareステートメントでAPIのインポートと、引数型の変換の宣言が必要になる。Declareステートメントの定義例を示して解説する。

```
Declare Function GetDeviceCaps Lib "gdi32" Alias "GetDeviceCaps"
(ByVal hdc As Long, ByVal nIndex As Long) As Long
```

上記ステートメントにおいて、GetDeviceCapsの部分が利用するAPIの関数名である。"gdi32"部分は、APIの関数GetDeviceCapsが格納されているDLL名である。WindowsのシステムAPIを呼び出す場合においては、kernel32.dll、gdi32.dll、user32.dllともに拡張子を除いた形で宣言する。その他のライブラリからAPIを呼び出すには、パスと拡張子（通常は.DLL）を含めた形式で宣言する。

Declareステートメント後半の（）内については引数の定義部分である。APIはC言語からの利用を前提に引数などが定義されているため、Visual Basicで利用するためには型変換が必要になる。次の表は主に利用される引数の型がC言語とVisual Basicでどうのように対応しているかを表わしている。変数型については、ここに掲載する以外にも多数存在する。詳細については『Windows SDK』を参照してもらいたい。

Declareステートメント末尾のAs以下は、API関数の戻り値の型を示している。

引数の種別	C言語での宣言	Visual Basicでの宣言
NULL	NULL	As Any
void	戻り値	–
voidへのポインタ	void*	As Any
ハンドル	hWnd,hDC,hMenu	By Val ～ As Integer
構造体へのポインタ	LPRECT	～ As Rect
整数	INT,UINT,WORD,BOOL	By Val ～ As Integer
整数へのポインタ	LPINT	～ As Integer
整数配列へのポインタ	――――	～ As Integer
長整数	DWORD,LONG	By Val ～ As Long
長整数へのポインタ	LPDWORD	～ As Long
文字列へのポインタ	LPSTR	By Val ～ As String

本書の「第2章　Win32 APIリファレンス」では、Visual Basicからでも利用できるAPIについて、Declare宣言を併記している。また、本書で取り扱っていないAPIについては、『Win32 SDK』や『Visual Basic プロフェッショナル版』以上に添付の『APIビューワー』を参照してもらいたい。

1.5.2 構造体と定数の定義

Declareステートメントで使用するAPIの宣言を終えたら、構造体とパラメータや戻り値として利用される定数値を定義する。

構造体はTypeステートメント、定数はConstステートメントを使いそれぞれの定義を行う。

本書の「第2章　Win32 APIリファレンス」で利用している構造体は「付録C　Visual Basic用構造体一覧」にTypeステートメントを含めて掲載している。また、定数については「付録D　Visual Basic用定数一覧」に掲載している。本書で扱っていない構造体や定数については、『Win32 SDK』に添付されている"windows.h"のヘッダファイルを参照するか、『Visual Basic プロフェッショナル版』以上に添付の『APIビューワー』を参照してもらいたい。

先の例を用いると、定数の宣言は次のようになる。

```
Const HORZRES = 8
Const VERTRES = 10
```

先の例では構造体を利用していないが、もし構造体を利用しているAPIであれば、呼び出しの前に次のように構造体の定義を行う。もちろん、構造体の中でさらに構造体へのポインタを参照している場合（次の例ではMSG構造体からPOINTAPI構造体を参照している）は、すべての構造体についての定義を行う。

```
Type MSG
    hwnd As Long
    message As Long
    wParam As Long
    lParam As Long
    time As Long
    pt As POINTAPI
End Type

Type POINTAPI
    x As Long
    y As Long
End Type
```

1.5.3 APIの呼び出し

Declareステートメントでの宣言と、構造体や定数の定義を終えると、APIをユーザー定義のファンクションやサブプロシージャと同様に利用できる。例では、Declareステートメントは、［Form］ウィンドウのオブジェクト：（General），プロシージャ：（declarations）に記述している。

```
Declare Function GetDeviceCaps Lib "gdi32" Alias "GetDeviceCaps"
(ByVal hdc As Long, ByVal nIndex As Long) As Long

Sub Form_load ()
   Const HORZRES = 8
   Const VERTRES = 10
   Show
   Print "現在のモニタ解像度は";
   Print GetDeviceCaps ( hDC, HORZRES ) ; "×";
   Print GetDeviceCaps ( hDC, VERTRES ) ; "ドットです";
End Sub
```

このサンプルでは、フォームのロード時にAPI関数GetDeviceCapsを利用して、現在のモニタ解像度をフォーム上に表示している。

Visual BasicでのAPI関数利用の際に注意しなければいけない点はAPI関数の呼び出しをおこなう時に変数値のチェックなどがおこなわれない点である。デバッグやトレース作業を進める時に注意したい。また取得したハンドルなどは忘れずに明示的な解放をする必要がある。

第2章

Win32 API リファレンス

2.1 リファレンスの見方

以降のリファレンスは、数千あるWin32 APIの中からよく使われるであろう300APIに絞り込み説明をしている。これら300のAPIは、下記のように機能ごとにまとめて掲載している。

2.2 Window Management／ウィンドウマネージメント

Window, Caret, Clipboard, Common Dialog Box, Cursor, Icon, Message and Message Queue, Mouse Input, Painting and Drawing, Rectangle, Timers, Window Class, Window Property

2.3 Window Controls／ウィンドウコントロール

Common Controls, Drag List Box, Property Sheet, Status Window Toolbar, Up-Down Control, Button, Combo Box, Dialog Box, Edit Control, List Box, Scroll Bar

2.4 Shell Features／シェル

Control Panel, File Installation Library, Screen Saver, Shell Library

2.5 Graphic Device Interface／GDIインターフェイス

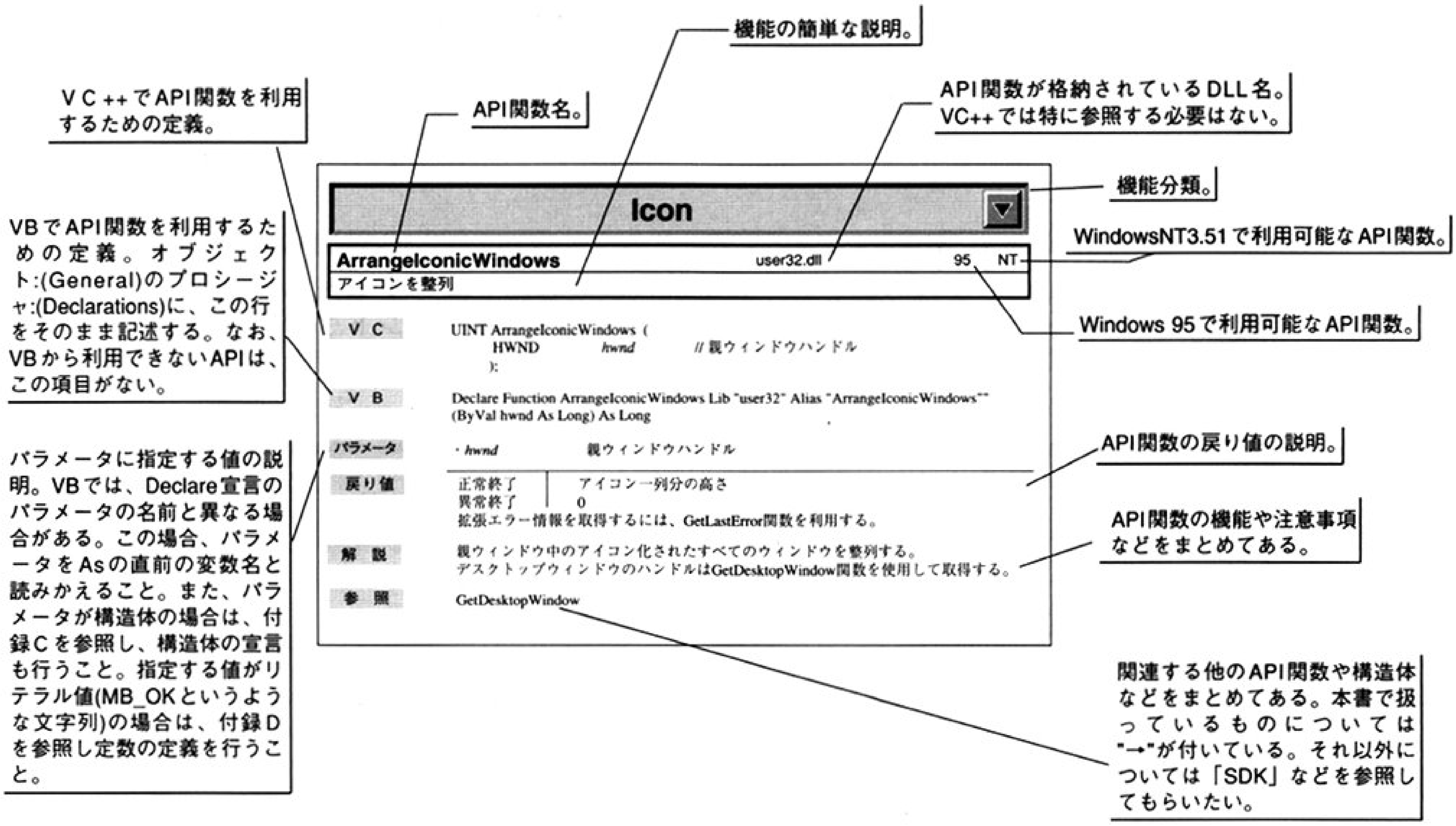
Bitmap, Brush, Clipping, Color, Coordinate Space and Transformation, Device Context, Filled Shape, Font and Text, Metafile, Path Region

2.6 System Service／システムサービス

Accessibility, Data Decompression Library, Error, File, Help Memory Management, Pipe, Process and Thread, Registory, Resource String Manipulation, Structured Exception Handling, System Information, Time, Window Station and Desktop

API関数の機能別の一覧やアルファベット順一覧は、それぞれ「付録A　機能別API一覧」「付録B　アルファベット順API一覧」にまとめた。
本書で扱っていないAPIについては「Win32 SDK」（英語）を参照してもらいたい。

次に、リファレンスの見方を説明する。



●Viausl C++ユーザーの方へ

V B 以外の項目を参照してもらいたい。API関数とその周辺機能に関して知識を有する場合は、V C の項目を参照するだけで十分その機能と用法が理解できるかと思われる。
なおDLL名は、Visual Basicユーザーのためのものである。VC++からAPI関数を呼び出す場合は、ヘッダファイルをインクルード（#include <windows.h>）するだけでよい。
詳細については、VC++に添付の「Windows SDKオンラインヘルプ」を合わせて参照してほしい。

●Visual Basicユーザーの方へ

V C 以外の項目を参照してもらいたい。VBから利用できないAPIは、V B の項目を記載していない。
パラメータ や 解説 では、VCの定義を元に説明を行っているので、パラメータ名が V B で定義したものと異なる場合がある。API関数を呼び出す際のパラメータの数や順番は、VC++／VBに違いはない。したがって、パラメータ のパラメータ名を、Declareステートメントで宣言したパラメータ名に読み換えていただきたい。
特に注意すべき点は、パラメータに構造体が指定されている場合は「付録C　Visual Basic用構造体定義一覧」を参照して、DeclareとともにTypeの宣言もしなければならない。また、構造体の中で構造体を指定している場合もあるので、注意していただきたい。
解説中のリテラル定数（"MB_YESNO"や"WM_CREATE"という文字列で表された定数）は「付録D　VisualBasic用定数一覧」を参照していただきたい。プログラム中では、あらかじめPublic Constステートメントで定数を定義し、パラメータに値を指定する場合は、リテラル定数を用いるとより見やすいプログラムとなる。もちろん、定数を直接指定してもかまわない。
Visual Basicのプロフェッショナル版、もしくはエンタープライズ版の方は、製品添付の「APIビューア」を利用すると、Declare宣言や構造体の定義、定数の定義が楽にできるので本書と合わせて利用するとよい。

2. 2 Window Management
ウィンドウマネージメント

ウィンドウのフレームをはじめとした基礎的な構成要素を操作するためのAPI関数群がウィンドウマネージメント関数である。フレームウィンドウの作成から破棄、マウスやアイコン、ダイアログメッセージなどの処理を行うための必須関数群である。

Window

AdjustWindowRect	user32.dll	95	NT
ウィンドウサイズを変更する			

V C

```
BOOL AdjustWindowRect (
    LPRECT  lprc,       //長方形の座標を示すRECT構造体へのポインタ
    DWORD   dwStyle,    //変換されるウィンドウのウィンドウスタイル
    BOOL    fMenu       //ウィンドウがメニューを持つかどうかを指定する
    );
```

V B

```
Declare Function AdjustWindowRect Lib "user32" Alias "AdjustWindowRectA" (lpRect
As RECT, ByVal dwStyle As Long, ByVal bMenu As Long) As Long
```

パラメータ

- *lprc*　クライアント長方形の座標を示すRECT構造体へのポインタである。
- *dwStyle*　変換されるクライアント長方形を含むウィンドウのウィンドウスタイル。

値	意味
WS_BORDER	境界を持つウィンドウ。
WS_CAPTION	タイトルバーを持つウィンドウ（暗黙にWS_BORDERスタイルを持つ）。WS_DLGFRAMEスタイルと一緒に使うことはできない。
WS_DLGFRAME	二重境界を持ち、タイトルを持たないウィンドウ。
WS_HSCROLL	水平スクロールバーを持つウィンドウ。
WS_THICKFRAME	ウィンドウのサイズ変更に使うことができる、太い枠を持つウィンドウ。
WS_VSCROLL	垂直スクロールバーを持つウィンドウ。

- *fMenu*　ウィンドウがメニューを持つ場合はTRUEを、持たない場合はFALSEを指定する。

戻り値

正常終了	TRUE
異常終了	FALSE

拡張エラー情報を取得するには、GetLastError関数を使う。

解　説

指定されたクライアント長方形のサイズに基づいて、ウィンドウ長方形に必要なサイズを計算する。このサイズをCreateWindow関数に引き渡すことで目的のクライアントサイズを持つウィンドウを作成する。

AdjustWindowRect関数は、クライアント領域のサイズを計算するとき、タイトルや境界の幅を考慮しない。タイトルや境界を持つウィンドウスタイルの場合、アプリケーションはAdjustWindowRect関数を呼び出した後でそれらのサイズを加算する必要がある。メニューバーが複数行に折り返している場合にも、2行目以降のサイズは考慮されない。

参　照

AdjustWindowRectEx,CreateWindowEx

CreateWindow	user32.dll	95	NT
ウィンドウの作成			

V　C

```
HWND CreateWindow (
    LPCTSTR   lpszClassName,   //NULLで終わる文字列へのポインタ
    LPCTSTR   lpszWindowName,  //NULLで終わる文字列へのポインタ
    DWORD     dwStyle,         //作成されるウィンドウのスタイル
    int       x,               //初期状態でのウィンドウのx座標の位置
    int       y,               //初期状態でのウィンドウのy座標の位置
    int       nWidth,          //ウィンドウの幅をデバイス単位で指定
    int       nHeight,         //ウィンドウの高さをデバイス単位で指定
    HWND      hwndParent,      //作成されるウィンドウの親ウィンドウを
                                 識別するハンドル
    HMENU     hmenu,           //ウィンドウスタイルに依存する子ウィン
                                 ドウID
    HANDLE    hinst,           //ウィンドウに関連付けられているモジュー
                                 ルインスタンス
    LPVOID    lpvParam,        //CREATESTRUCT構造体を通じてウィンド
                                 ウに渡される値を指すポインタ
    );
```

V　B

Declare Function CreateWindow Lib "user32" Alias "CreateWindowA" (ByVal lpClassName As String, ByVal lpWindowName As String, ByVal dwStyle As Long, ByVal x As Long, ByVal y As Long, ByVal nWidth As Long, ByVal nHeight As Long, ByVal hWndParent As Long, By Val hMenu As Long, By Val hInstance As Long, lpParam As Any) As Long

パラメータ

- *lpszClassName*　NULLで終わる文字列を指すポインタか、または整数アトムである。このパラメータがアトムの場合、以前のGlobalAddAtom関数の呼び出しによって作成されたグローバルアトムでなければならない。アトムは、0xC000未満の16ビット値で、*lpszClassName*の下位ワードになければならない。上位ワードは0になる。*lpszClassName*が文字列を指す場合、ウィンドウクラスの名前を指定する。クラス名には、RegisterClass関数で登録された名前や、定義済みのコントロールクラス名が指定できる。
- *lpszWindowName*　ウィンドウ名を表すNULLで終わる文字列を指すポインタである。
- *dwStyle*　作成されるウィンドウのスタイルを指定する。指定できる値は、AjustWindowRectを参照。
- *x*　初期状態でのウィンドウのx座標の位置を指定する。オーバーラップウィンドウやポップアップウィンドウでは、*x*パラメータはウィンドウの左上隅の初期x座標になる。子ウィンドウでは、親ウィンドウのクライアント領域の左上隅に相対的な子ウィンドウの左上隅のx座標になる。この値にCW_USEDEFAULTを指定すると、Windowsはウィンドウの左上隅の位置としてデフォルトの位置を選択して、*y*パラメータを無視する。CW_USEDEFAULTはオーバーラップウィンドウに対してだけ有効である。ポップアップウィンドウや子ウィンドウに対してCW_USEDEFAULTを指定した場合、*x*パラメータと*y*パラメータはともに0に設定される。
- y　初期状態でのウィンドウのy座標の位置を指定する。オーバーラップウィンドウやポップアップウィンドウでは、*y*パラメータはウィンドウの左上隅の初期y座標になる。子ウィンドウでは、親ウィンドウのクライアント領域内の左上隅に相対的な子ウィンドウの左上隅の初期y座標になる。リストボックスでは、親ウィンドウのクライアント領域の左上隅に相対的なリストボックスのクライアント領域の左上隅の初期y座標になる。オーバーラップウィンドウをWS_VISIBLEスタイルで作成して*x*パラメータをCW_USEDEFAULTに設定した場合、Windowsは*y*パラメータを無視する。
- *nWidth*　ウィンドウの幅をデバイス単位で指定する。オーバーラップウィンドウでは、*nWidth*パラメータはスクリーン座標単位でのウィンドウの幅、またはCW_USEDEFAULTになる。*nWidth*がCW_USEDEFAULTのとき、Windowsはウィンドウに対してデフォルトの幅と高さを選択する。デフォルトの幅は初期状態でのx座標位置からスクリーンの右端までの長さ、デフォルトの高さは初期状態でのy座標位置からアイコン領域の上端までの長さである。CW_USEDEFAULTはオーバーラップウィンドウに対してだけ有効である。ポップアップウィンドウや子ウィン

ドウに対してCW_USEDEFAULTを指定した場合、*nWidth*と*nHeight*はともに0に設定される。

- *nHeight* ウィンドウの高さをデバイス単位で指定する。オーバーラップウィンドウでは、*nHeight*パラメータはスクリーン座標単位でのウィンドウの高さになる。*nWidth*パラメータがCW_USEDEFAULTに設定された場合、Windowsは*nHeight*を無視する。
- *hwndParent* 作成されるウィンドウの親ウィンドウまたはオーナーウィンドウを識別する。子ウィンドウやオーナー付きウィンドウを作成するときには、有効なウィンドウハンドルを提供しなければならない。子ウィンドウはその親ウィンドウのクライアント領域に置かれることになる。オーナー付きウィンドウとは、そのオーナーウィンドウが破棄されるときに破棄され、オーナーウィンドウがアイコン化されるときに非表示となるオーバーラップウィンドウのことで、常にオーナーウィンドウの上に表示される。このパラメータは、*dwStyle*パラメータがWS_CHILDスタイルを含むときは有効なハンドルを指定しなければならないが、*dwStyle*がWS_POPUPスタイルを含むときには省略可能である。
- *hmenu* メニューの識別、またはウィンドウのスタイルに依存する子ウィンドウIDを指定する。オーバーラップウィンドウまたはポップアップウィンドウでは、*hmenu*パラメータはウィンドウで使われるメニューを識別する。クラスメニューを使うときにはNULLが指定できる。子ウィンドウの場合には、子ウィンドウIDで、親ウィンドウにイベントを通知するダイアログボックスコントロールが使う整数値になる。子ウィンドウIDはアプリケーションにより決められ、同じ親ウィンドウを持つすべての子ウィンドウに対して一意でなければならない。
- *hinst* そのウィンドウに関連付けられているモジュールのインスタンスを識別する。
- *lpvParam* WM_CREATEメッセージの*lParam*パラメータが参照するCREATESTRUCT構造体を通じてウィンドウに渡される値を指すポインタである。アプリケーションがCreateWindow関数を呼び出してマルチドキュメントインターフェイス（MDI）のクライアントウィンドウを作成するときには、*lpvParam*パラメータはCLIENTCREATESTRUCT構造体へのポインタでなければならない。

戻り値

正常終了	ウィンドウハンドル
異常終了	NULL

拡張エラー情報を取得するには、GetLastError関数を利用する。

解　説

オーバーラップウィンドウ、ポップアップウィンドウ、子ウィンドウを作成する。CreateWindow関数は、ウィンドウクラス、ウィンドウタイトル、ウィンドウスタイ

ル、ウィンドウの初期状態での位置とサイズを指定する。また、親ウィンドウまたはオーナーウィンドウ、およびメニューも指定可能である。
CreateWindow関数は、制御を戻す前にWM_CREATEメッセージをウィンドウプロシージャに送る。

参　照

→CreateDialog,CreateWindowEx,→DialogBox,→MessageBox,→CREATESTRUCT,
→CLIENTCREATESTRUCT

DestroyWindow	user32.dll	95	NT
ウィンドウの破棄			

V　C

```
BOOL DestroyWindow (
        HWND          hwnd        //破棄されるウィンドウを識別する
        );
```

V　B

Declare Function DestroyWindow Lib "user32" Alias "DestroyWindow" (ByVal hwnd As Long) As Long

パラメータ

・*hwnd*　　破棄されるウィンドウを識別する。

戻り値

正常終了	TRUE
異常終了	FALSE

拡張エラー情報を取得するには、GetLastError関数を利用する。

解　説

指定されたウィンドウを破棄する。この関数は、WM_DESTROYメッセージおよびWM_NCDESTROYメッセージをウィンドウに送ってウィンドウを非アクティブ化し、入力フォーカスを取り除く。また、この関数は、ウィンドウメニューの破棄、スレッドのメッセージキューのフラッシュ、タイマの破棄、クリップボードの所有権の除去、クリップボードビューアチェインの切断をそれぞれ実行する。
スレッドは、DestroyWindow関数を使って、別のスレッドによって作成されたウィンドウを破棄することはできない。破棄されるウィンドウが子ウィンドウで、WS_EX_NOPARENTNOTIFYスタイルがセットされていないときは、WM_PARENTNOTIFYメッセージが親ウィンドウに送られる。

参　照

→CreateDialog,→CreateWindow,CreateWindowEx,→WM_DESTROY,
→WM_NCDESTROY

GetWindow	user32.dll	95	NT
ウィンドウのハンドルの取得			

V C

```
BOOL GetWindow (
    HWND   hWnd,   //ウィンドウハンドルを取得するために利用する関連ウィ
                     ンドウハンドル
    UINT   uCmd    //hWndパラメータで示されるウィンドウとの関係
    );
```

V B

Declare Function GetWindow Lib "user32" Alias "GetWindow" (ByVal hwnd As Long, By Val wCmd As Long) As Long

パラメータ

- *hWnd*　ウィンドウハンドルを取得するために利用する関連ウィンドウハンドル。
- *uCmd*　パラメータで示されるウィンドウとの関係。*uCmd*パラメータには以下の値のいずれかを指定する。

値	意味
GW_CHILD	最前面にある子ウィンドウを識別するハンドルを取得する。
GW_HWNDFIRST	最上位ウィンドウを識別するハンドルを返す。指定されたウィンドウが子ウィンドウの場合には兄弟ウィンドウのハンドルを返す。
GW_HWNDLAST	最下位ウィンドウを識別するハンドルを返す。指定されたウィンドウが子ウィンドウの場合には兄弟ウィンドウのハンドルを返す。
GW_HWNDNEXT	指定されたウィンドウの下位ウィンドウを識別するハンドルを返す。最上位ウィンドウが指定された場合には指定されたウィンドウのハンドルを返す。子ウインドウが指定された場合には兄弟ウィンドウのハンドルが返される。
GW_HWNDPREV	指定されたウィンドウの上位ウィンドウを識別するハンドルを返す。最上位ウィンドウが指定された場合には指定されたウィンドウのハンドルを返す。子ウインドウが指定された場合には兄弟ウィンドウのハンドルが返される。
GW_OWNER	オーナーウィンドウを識別するウィンドウハンドルを返す。

戻り値

正常終了	ウィンドウハンドル
異常終了	FALSE

拡張エラー情報を取得するには、GetLastError関数を利用する。

解　説

指定されたウィンドウのウィンドウハンドルを取得する。*hWnd*パラメータで指定されたウィンドウを基準に*uCmd*パラメータの関係を満たす最初のウィンドウハンドルを検索する。

参　照

GetActiveWindow,GetNextWindow,GetTopWindow

GetWindowRect	――――――	95	NT
ウィンドウ全体のサイズを取得			

V　C

```
BOOL GetWindowRect(
    HWND      hWnd,     //ウィンドウのハンドル
    LPRECT    lpRect    //RECT構造体へのポインタ
    );
```

パラメータ

・*hWnd*　ウィンドウを識別するハンドル。

・*lpRect*　ウィンドウの左上から右下コーナーのスクリーン座標を受け取るRECT構造体へのポインタ。

戻り値

正常終了	TRUE
異常終了	FALSE

拡張エラー情報はGetLastError関数で取得する。

参　照

GetClientRect,→RECT

GetWindowText	――――――	95	NT
ウィンドウのタイトルを取得			

V　C

```
int GetWindowText(
    HWND     hWnd,        //ウィンドウもしくはコントロールオブジェクトの
                           ハンドル
    LPTSTR   lpString,    //文字列を格納するバッファへのポインタ
    int      nMaxCount    //受け取る文字列の最大長
    );
```

パラメータ

・*hWnd*　ウィンドウもしくはコントロールオブジェクトのハンドルを指定する。

・*lpString*　テキストを格納するためのNULLで終わる文字列へのポインタ。

・*nMaxCount*　*lpString*にコピーされるテキストの最大長を指定する。受け取った文字列がこの値よりも大きい場合には切りつめられる。

戻り値

正常終了	NULLで終わる文字を含まない文字列の長さ
異常終了	0

拡張エラー情報はGetLastError関数で取得する。

解　説

この関数は、WM_GETTEXTメッセージを指定されたウィンドウまたはコントロールに送る。この関数でエディットコントロールのテキストを取り出すことはできない。

参　照

GetWindowTextLength,→SetWindowText,→WM_GETTEXT

MoveWindow	user32.dll	95	NT
ウィンドウのサイズと位置を変更			

V C

```
BOOL MoveWindow(
    HWND    hWnd,     //ウィンドウを識別するハンドル
    int     X,        //原点のX座標
    int     Y,        //原点のY座標
    int     nWidth,   //ウィンドウの幅
    int     nHeight,  //ウィンドウの高さ
    BOOL    bRepaint  //再描画フラグ
    );
```

V B

```
Declare Function MoveWindow Lib "user32" Alias "MoveWindow" (ByVal hwnd As Long, ByVal x As Long, ByVal y As Long, ByVal nWidth As Long, ByVal nHeight As Long, ByVal bRepaint As Long) As Long
```

パラメータ

- *hWnd*　ウィンドウを識別するハンドルを指定する。
- *X*　ウィンドウの左上のX座標を指定する。
- *Y*　ウィンドウの左上のY座標を指定する。
- *nWidth*　ウィンドウの幅を指定する。
- *nHeight*　ウィンドウの高さを指定する。
- *bRepaint*　ウィンドウを再描画するかどうかを指定する。パラメータがTRUEの場合、ウィンドウはWM_PAINTメッセージを受け取る。パラメータがFALSEの場合には、いかなる再描画も実行されない。

戻り値

正常終了	TRUE
異常終了	FALSE

解説

*bRepaint*パラメータがTRUEの場合には、WindowsはウィンドウをC動かした直後にWM_PAINTメッセージをOSに送る（すなわち、MoveWindowファンクションコールUpdateWindow関数）。*bRepaint*がFALSEの場合には、Windowsは、WM_PAINTメッセージをウィンドウと関連したメッセージキューに送る。

参照

SetWindowPos,UpdateWindow,→WM_GETMINMAXINFO,→WM_PAINT

SetWindowText	user32.dll	95	NT
ウィンドウのタイトルやテキストを設定			

V C

```
BOOL SetWindowText(
    HWND      hWnd,      //ウィンドウもしくはコントロールのハンドル
    LPCTSTR   lpString   //テキストを格納したNULLで終わる文字列へのポインタ
    );
```

V B

Declare Function SetWindowText Lib "user32" Alias "SetWindowTextA" (ByVal hwnd As Long, ByVal lpString As String) As Long

パラメータ

- *hWnd*　タイトルテキストを変更したいウィンドウのハンドルを指定する。
- *lpString*　テキストを格納したNULLで終わる文字列へのポインタを指定する。

戻り値

正常終了	TRUE
異常終了	FALSE

拡張エラー情報はGetLastError関数で取得する。

解　説

SetWindowText関数は、WM_SETTEXTメッセージを指定されたウィンドウもしくはコントロールに送る。SetWindowText関数はタブ文字を認識しない（ASCIIコード0x09）。タブ文字は垂直のバー（|）文字として表示される。

Caret

CreateCaret	user32.dll	95	NT
キャレットの作成			

V C

```
void CreateCaret (
    HWND      hwnd,      //新しいキャレットを所有するウィンドウハンドル
    HBITMAP   hbmp,      //キャレットの形状を定義したビットマップハンドル
    int       nWidth,    //キャレットの幅を論理単位で指定
    int       nHeight    //キャレットの高さを論理単位で指定
    );
```

V B

Declare Function CreateCaret Lib "user32" Alias "CreateCaret" (ByVal hwnd As Long, ByVal hBitmap As Long, ByVal nWidth As Long, ByVal nHeight As Long) As Long

パラメータ

- *hwnd*　新しいキャレットを所有するウィンドウを識別する。
- *hbmp*　キャレットの形状を定義したビットマップを識別する。このパラメータがNULLの場合にはキャレットは標準表示になり、1の場合には灰色表示となる。

・*nWidth* キャレットの幅を論理単位で指定する。このパラメータがNULLの場合は、キャレットの幅はシステム定義のウィンドウ境界の幅に設定される。*hbmp*がビットマップのハンドルの場合、CreateCaretはこのパラメータを無視する。

・*nHeight* キャレットの高さを論理単位で指定する。このパラメータがNULLの場合は、キャレットの高さはシステム定義のウィンドウ境界の高さに設定される。*hbmp*がビットマップのハンドルの場合、CreateCaretはこのパラメータを無視する。

戻り値

正常終了	TRUE
異常終了	FALSE

拡張エラー情報を取得するには、GetLastError関数を利用する。

解説

システムキャレットの形状を新たに作成し、指定されたウィンドウにそのキャレットの所有権を与える。キャレットの形状は、線やブロック、またはビットマップにすることが可能である。

*nWidth*パラメータと*nHeight*パラメータは、キャレットの幅と高さを論理単位で指定するが、正確な幅と高さは、ウィンドウのマッピングモードに依存する。

参照

→CreateBitmap,→CreateDIBitmap,DestroyCaret,→GetSystemMetrics,HideCaret,→LoadBitmap,ShowCaret

GetCaretPos	user32.dll	95	NT
キャレットの現在位置を取得			

VC

```
BOOL GetCaretPos (
    LPPOINT    lpPoint    //クライアント座標を受け取るPOINT構造体への
                            ポインタ
    );
```

VB

```
Declare Function GetCaretPos Lib "user32" Alias "GetCaretPos" (lpPoint As POINTAPI) As Long
```

パラメータ

・*lpPoint* キャレットの現在位置を受け取るPOINT構造体へのポインタ。

戻り値

正常終了	TRUE
異常終了	FALSE

拡張エラー情報を取得するには、GetLastError関数を利用する。

解説

クライアント座標で示されたキャレットの現在の位置を、指定されたPOINT構造体にコピーする。キャレットの位置は、常にキャレットのあるウィンドウのクライアント座標として与えられる。

参照

SetCaretPos,→POINT

HideCaret	user32.dll	95	NT
指定されたウィンドウからキャレットを削除			

V　C

```
BOOL HideCaret(
    HWND        hWnd    //ウィンドウを識別するハンドル
    );
```

V　B

Declare Function HideCaret Lib "user32" Alias "HideCaret" (ByVal hwnd As Long) As Long

パラメータ

・*hWnd*　　キャレットを削除するウィンドウを識別するハンドル。

戻り値

正常終了	TRUE
異常終了	FALSE

拡張エラー情報を取得するには、GetLastError関数を利用する。

解　説

指定されたウィンドウがキャレットを所有している場合には、HideCaretはキャレットを隠す。指定されたウィンドウがキャレットを所有していない場合には、HideCaret関数は何もせず、FALSEを戻す。
HideCaret関数が5回呼ばれた場合には、再度キャレットを表示するためにShowCaret関数を5回呼ばなければならない。

参　照

→CreateCaret,DestroyCaret,→GetCaretPos,SetCaretPos,ShowCaret

Clipboard

CloseClipboard	user32.dll	95	NT
クリップボードをクローズ			

V　C

```
BOOL CloseClipboard ( ) ;
```

V　B

Declare Function CloseClipboard Lib "user32" Alias "CloseClipboard" () As Long

戻り値

正常終了	TRUE
異常終了	FALSE

拡張エラー情報を取得するには、GetLastError関数を利用する。

解　説

クリップボードをクローズする。
ウィンドウがクリップボードの参照または変更を終了したときには、CloseClipboard関数を呼び出すことでクリップボードをクローズしなければならない。これにより、ほかのアプリケーションがクリップボードにアクセス可能となる。

参　照

DuplicateHandle,GetOpenClipboardWindow,OpenClipboard

EmptyClipboard	user32.dll	95	NT
クリップボードを空にする			

V C

BOOL EmptyClipboard ( ) ;

V B

Declare Function EmptyClipboard Lib "user32" Alias "EmptyClipboard" () As Long

戻り値

正常終了	TRUE
異常終了	FALSE

拡張エラー情報を取得するには、GetLastError関数を利用する。

解 説

クリップボードの内容を空にし、クリップボード内のデータを識別するハンドルを解放する。そして、現在クリップボードをオープンしているウィンドウに、クリップボードの所有権を与える。
アプリケーションは、EmptyClipboard関数を呼び出す前にOpenClipboard関数を使ってクリップボードをオープンしなければならない。クリップボードをオープンするときにアプリケーションがNULLのウィンドウハンドルを指定する場合、EmptyClipboard関数は正常に終了するが、クリップボードのオーナーはNULLに設定される。

参 照

OpenClipboard,SetClipboardData,→WM_DESTROYCLIPBOARD

GetClipboardData	user32.dll	95	NT
クリップボードデータの取得			

V C

```
HANDLE GetClipboardData (
    UINT        uFormat    //クリップボード形式
    );
```

V B

Declare Function GetClipboardData Lib "user32" Alias "GetClipboardDataA" (ByVal wFormat As Long) As Long

パラメータ

・*uFormat*　　クリップボード形式を指定する。

戻り値

正常終了	クリップボードオブジェクトへのハンドル
異常終了	NULL

拡張エラー情報を取得するには、GetLastError関数を利用する。

解 説

指定された形式を持つクリップボードのデータを取得する。クリップボードはあらかじめオープンされていなければならない。
EnumClipboardFormats関数を使用して、あらかじめ有効なデータ形式を列挙しておくことも可能である。

参 照

EnumClipboardFormats,SetClipboardData

OpenClipboard	user32.dll	95	NT
クリップボードをオープンする			

V C

```
BOOL OpenClipboard(
    HWND    hWndNewOwner  //ウィンドウを識別するハンドル
    );
```

V B

Declare Function OpenClipboard Lib "user32" Alias "OpenClipboard" (ByVal hwnd As Long) As Long

パラメータ

・*hWndNewOwner*　　オープンするクリップボードと結び付けられるウィンドウを識別するハンドルを指定する。パラメータにNULLを指定すると、現在のタスクとクリップボードが結び付けられる。

戻り値

正常終了	クリップボードオブジェクトへのハンドル
異常終了	NULL

拡張エラー情報を取得するには、GetLastError関数を利用する。

解　説

別のウィンドウがクリップボードを所有している場合には、OpenClipboard関数は失敗する。アプリケーションは、必ずCloseClipboard関数でオープンしたクリップボードを閉じる必要がある。

参　照

→CloseClipboard,→EmptyClipboard

Common Dialog Box

ChooseColor	comdlg32.dll	95	NT
色選択のダイアログを作成			

V C

```
BOOL ChooseColor (
    LPCHOOSECOLOR    lpcc    //CHOOSECOLOR構造体へのポインタ
    );
```

V B

Declare Function ChooseColor Lib "comdlg32.dll" Alias "ChooseColorA" (pChoosecolor As CHOOSECOLOR) As Long

パラメータ

・*lpcc* 呼び出し時は、ダイアログボックスを初期化するために必要な情報を格納するCHOOSECOLOR構造体へのポインタである。ChooseColor関数が制御を戻すときは、この構造体にユーザーが選択した色に関する情報が格納される。

戻り値

ユーザーが色を選択した場合	TRUE
それ以外の場合	FALSE

解　説

ユーザーが色を選択できるシステム定義のダイアログボックスを作成する。
このダイアログボックスでは、カラーパレットをサポートしない。ダイアログボックスが提供する色の選択肢は、システムカラーと、システムカラーのディザカラーに限られる。

参　照

CommDlgExtendedError,→WM_CTLCOLORDLG

ChooseFont	comdlg32.dll	95	NT
フォント選択のダイアログを作成			

V C

```
BOOL ChooseFont (
   LPCHOOSEFONT   lpcf   //CHOOSEFONT構造体へのポインタ
   );
```

V B

```
Declare Function ChooseFont Lib "comdlg32.dll" Alias "ChooseFontA" (pChoosefont
As CHOOSEFONT) As Long
```

パラメータ

・*lpcf* 呼び出し時は、ダイアログボックスを初期化するために必要な情報を格納するCHOOSEFONT構造体へのポインタである。ChooseFontが制御を戻すときには、この構造体にユーザーが選択したフォントに関する情報が格納される。

戻り値

正常終了	TRUE
異常終了	FALSE

解　説

ユーザーがフォントを選択できるシステム定義のダイアログボックスを作成する。
フック関数がWM_CTLCOLORDLGメッセージを処理する場合、コントロールの背景を描画するために使われるブラシのハンドルを返さなければならない。

参　照

CommDlgExtendedError,WM_CHOOSEFONT_GETLOGFONT,
WM_CTLCOLORDLG,→CHOOSEFONT

FindText	comdlg32.dll	95	NT
テキスト検索のダイアログを作成			

V C

```
HWND FindText(
    LPFINDREPLACE  lpfr   //初期設定データを格納したFINDREPLACE構造
                            体へのポインタ
    );
```

V B

Declare Function FindText Lib "comdlg32.dll" Alias "FindTextA " (pFindreplace As FINDREPLACE) As Long

パラメータ

・*lpfr*　ダイアログボックスを初期設定するFINDREPLACE構造体へのポインタを指定する。

戻り値

正常終了	ダイアログボックスのウィンドウハンドル
異常終了	NULL

拡張エラー情報はGetLastErrorで取得する。GetLastError関数は次の値を返す。

CDERR_FINDRESFAILURE,CDERR_INITIALIZATION,
CDERR_LOCKRESFAILURE,CDERR_LOADRESFAILURE,
CDERR_LOADSTRFAILURE,CDERR_MEMALLOCFAILURE,
CDERR_MEMLOCKFAILURE,CDERR_NOHINSTANCE,CDERR_NOHOOK,
CDERR_NOTEMPLATE,CDERR_STRUCTSIZE,FRERR_BUFFERLENGTHZERO

解　説

FindText関数は検索操作は実行しない。
アプリケーションは、呼び出し時にRegisterWindowMessage関数に「commdlg_FindReplace」ストリングを指定し、識別子をこれらのメッセージに登録できる。

参　照

CommDlgExtendedError,→FINDREPLACE,IsDialogMessage,RegisterWindowMessage,
→ReplaceText,→WM_CTLCOLORDLG,

GetFileTitle	comdlg32.dll	95	NT
ファイル名を取得			

V C

```
short GetFileTitle(
        LPCTSTR  lpszFile,   //フルパス名を格納する文字列へのポインタ
        LPTSTR   lpszTitle,  //ファイル名を受け取るための文字列へのポインタ
        WORD     cbBuf       //ファイル名を受け取るバッファのサイズ
        );
```

V B

Declare Function GetFileTitle Lib "comdlg32.dll" Alias "GetFileTitleA" (ByVal lpszFile As String, ByVal lpszTitle As String, ByVal cbBuf As Integer) As Integer

パラメータ

- *lpszFile*　ファイルのフルパス名を指定する文字列へのポインタ。
- *lpszTitle*　ファイル名を格納するための文字列へのポインタ。
- *cbBuf*　*lpszTitle*に格納されるファイル名のバッファサイズを指定する。

戻り値

正常終了	0
ファイル名が無効	負の整数
*lpszTitle*の値が小さい	正の整数

解　説

*lpszFile*パラメータが次の文字列の場合には、GetFileTitle関数はエラーを返す。

- 空の文字列
- 「*」,「[」,「]」のいずれかの文字が含まれる場合
- 「:」,「/」,「\」のいずれかの文字で終わる場合
- *cbBuf*で指定されたサイズよりも返されるファイル名の文字列が大きい場合
- 不正な文字が指定された場合

バッファサイズにはNULLで終わる文字列を含む長さを指定する。

参　照

→GetOpenFileName,→GetSaveFileName

GetOpenFileName	comdlg32.dll	95	NT
ファイルを開くダイアログボックスの作成			

V C

```
BOOL GetOpenFileName(
        LPOPENFILENAME      lpofn    //ダイアログボックス初期化用のOPEN
                                       FILENAME構造体へのポインタ
        );
```

V B

Declare Function GetOpenFileName Lib "comdlg32.dll" Alias "GetOpenFileNameA" (pOpenfilename As OPENFILENAME) As Long

パラメータ

- *lpofn*　ダイアログボックス初期化用のOPENFILENAME構造体へのポインタを指定する。

戻り値

ユーザーがファイルを選択	TRUE
それ以外	FALSE

拡張エラー情報はCommDlgExtendedErrorで取得する。CommDlgExtendedErrorは次のコードのいずれかを返す。

CDERR_FINDRESFAILURE,CDERR_INITIALIZATION,
CDERR_LOCKRESFAILURE,CDERR_LOADRESFAILURE,
CDERR_LOADSTRFAILURE,CDERR_MEMALLOCFAILURE,
CDERR_MEMLOCKFAILURE,CDERR_NOHINSTANCE,CDERR_NOHOOK,
CDERR_NOTEMPLATE,CDERR_STRUCTSIZE,FNERR_BUFFERTOOSMALL,
FNERR_INVALIDFILENAME,FNERR_SUBCLASSFAILURE

解説

フック関数（OPENFILENAME構造体のlpfnHookメンバによって示される）が、WM_CTLCOLORBTN、WM_CTLCOLORDLG、WM_CTLCOLORLISTBOX、WM_CTLCOLORMSGBOX、WM_CTLCOLORSCROLLBAR、またはWM_CTLCOLORSTATICメッセージを処理する場合、GetOpenFileNameはコントロール背景を塗りつぶすために使用されるブラシのハンドルを戻さなければならない。

参照

CommDlgExtendedError,→EndDialog,→GetSaveFileName,→OPENFILENAME

GetSaveFileName　comdlg32.dll　95　NT

名前を付けて保存のダイアログボックスを作成

VC

```
BOOL GetSaveFileName(
    LPOPENFILENAME      lpofn    //OPENFILENAME構造体へのポインタ
    );
```

VB

```
Declare Function GetSaveFileName Lib "comdlg32.dll" Alias "GetSaveFileNameA"
(pOpenfilename As OPENFILENAME) As Long
```

パラメータ

- *lpofn*　ダイアログボックス初期化用のOPENFILENAME構造体へのポインタを指定する。

戻り値

ユーザーがファイルを保存	TRUE
それ以外	FALSE

拡張エラー情報はCommDlgExtendedErrorで取得する。CommDlgExtendedErrorは次のコードのいずれかを返す。

CDERR_FINDRESFAILURE,CDERR_INITIALIZATION,CDERR_LOCKRES
FAILURE,CDERR_LOADRESFAILURE,CDERR_LOADSTRFAILURE,
CDERR_MEMALLOCFAILURE,CDERR_MEMLOCKFAILURE,
CDERR_NOHINSTANCE,CDERR_NOHOOK,CDERR_NOTEMPLATE,

CDERR_STRUCTSIZE,FNERR_BUFFERTOOSMALL,
FNERR_INVALIDFILENAME,FNERR_SUBCLASSFAILURE

解 説

フック関数（OPENFILENAME構造体のlpfnHookメンバによって示される）が、WM_CTLCOLORBTN、WM_CTLCOLORDLG、WM_CTLCOLORLISTBOX、WM_CTLCOLORMSGBOX、WM_CTLCOLORSCROLLBAR、またはWM_CTLCOLORSTATICメッセージを処理する場合、GetOpenFileNameはコントロール背景を塗りつぶすために使用されるブラシのハンドルを戻さなければならない。

参 照

CommDlgExtendedError,→GetOpenFilename,→OPENFILENAME,
→WM_CTLCOLORDLG

PageSetupDlg	comdlg32.dll	95	NT
ページ設定のダイアログボックスを作成			

V C

```
BOOL PageSetupDlg(
    LPPAGESETUPDLG      lppsd   //PAGESETUPDLG構造体へのポインタ
    );
```

V B

Declare Function PageSetupDlg Lib "comdlg32.dll" Alias "PageSetupDlgA" (pPagesetupdlg As PAGESETUPDLG) As Long

パラメータ

・*lppsd*　ダイアログボックス初期化用のPAGESETUPDLG構造体へのポインタを指定する。ダイアログボックスを初期設定するFINDREPLACE構造体へのポインタを指定する。

戻り値

ページ属性を設定変更した場合	TRUE
それ以外	FALSE

拡張エラー情報はCommDlgExtendedErrorで取得する。

参 照

CommDlgExtendedError,→PAGESETUPDLG

PrintDlg	comdlg32.dll	95	NT
印刷ダイアログボックスを作成			

V C

```
BOOL PrintDlg(
    LPPRINTDLG    lppd        //PRINTDLG構造体へのポインタ
    );
```

V B

Declare Function PrintDlg Lib "comdlg32.dll" Alias "PrintDlgA" (pPrintdlg As PRINTDLG) As Long

パラメータ

・*lppd*　ダイアログボックス初期化用のPRINTDLG構造体へのポインタを指定する。

戻り値

ダイアログボックスがOKボタンにより正常に終了	TRUE
それ以外	FALSE

拡張エラー情報はCommDlgExtendedErrorで取得する。CommDlgExtendedErrorは次のコードのいずれかを返す。

CDERR_FINDRESFAILURE,CDERR_INITIALIZATION,
CDERR_LOADRESFAILURE,CDERR_LOADSTRFAILURE,
CDERR_LOCKRESFAILURE,CDERR_MEMALLOCFAILURE,
CDERR_MEMLOCKFAILURE,CDERR_NOHINSTANCE,CDERR_NOHOOK,
CDERR_NOTEMPLATE,CDERR_STRUCTSIZE,PDERR_CREATEICFAILURE,
PDERR_DEFAULTDIFFERENT,PDERR_DNDMMISMATCH,
PDERR_GETDEVMODEFAIL,PDERR_INITFAILURE,PDERR_LOADDRVFAILURE,
PDERR_NODEFAULTPRN,PDERR_NODEVICES,PDERR_PARSEFAILURE,
PDERR_PRINTERNOTFOUND,PDERR_RETDEFFAILURE

解　説

フック関数（PRINTDLG構造体のlpfnPrintHookメンバによって示される）が、WM_CTLCOLORDLGメッセージを処理する場合、GetOpenFileNameはコントロール背景を塗りつぶすために使用されるブラシのハンドルを戻さなければならない。

参　照

CommDlgExtendedError,→CreateDC,DOCINFO,→PRINTDLG,StartDoc,
→WM_CTLCOLORDLG

ReplaceText	comdlg32.dll	95	NT
テキストの置換ダイアログボックスの作成			

V C

```
HWND ReplaceText(
        LPFINDREPLACE        lpfr     //FINDREPLACE構造体へのポインタ
        );
```

V B

Declare Function ReplaceText Lib "comdlg32.dll" Alias "ReplaceTextA" (pFindreplace As FINDREPLACE) As Long

パラメータ

・*lpfr*　　ダイアログボックス初期化用のFINDREPLACE構造体へのポインタを指定する。

戻り値

正常終了	置換ダイアログボックスのウィンドウハンドル
異常終了	NULL

拡張エラー情報はCommDlgExtendedError関数で取得する。CommDlgExtendedError関数は次のコードを返す。

CDERR_FINDRESFAILURE,CDERR_INITIALIZATION,
CDERR_LOADRESFAILURE,CDERR_LOADSTRFAILURE,
CDERR_LOCKRESFAILURE,CDERR_MEMALLOCFAILURE,
CDERR_MEMLOCKFAILURE,CDERR_NOHINSTANCE,CDERR_NOHOOK,
CDERR_NOTEMPLATE,CDERR_STRUCTSIZE,FRERR_BUFFERLENGTHZERO

解 説

ReplaceText関数では、実際に置換は行わない。
アプリケーションは、呼び出し時にRegisterWindowMessage関数に「commdlg_FindReplace」ストリングを指定して識別子をこれらのメッセージに登録できる。

参 照

CommDlgExtendedError,→FINDREPLACE,IsDialogMessage,RegisterWindowMessage,→WM_CTLCOLORDLG

Cursor

CreateCursor	user32.dll	95	NT
マウスカーソルの作成			

VC

```
HCURSOR CreateCursor (
    HINSTANCE    hinst,          //マウスカーソルを作成するアプリケーション
                                   のインスタンス
    int          xHotSpot,       //マウスカーソルのホットスポットの水平位置
    int          yHotSpot,       //マウスカーソルのホットスポットの垂直位置
    int          nWidth,         //マウスカーソルの幅をピクセル単位で指定
    int          nHeight,        //マウスカーソルの高さをピクセル単位で指定
    CONST VOID   *pvANDplane,    //マウスカーソルANDマスク用ビット値を表
                                   すバイト配列へのポインタ
    CONST VOID   *pvXORplane     //マウスカーソルXORマスク用ビット値を表
                                   すバイト配列へのポインタ
);
```

VB

```
Declare Function CreateCursor Lib "user32" Alias "CreateCursor" (ByVal hInstance As Long, ByVal nXhotspot As Long, ByVal nYhotspot As Long, ByVal nWidth As Long, ByVal nHeight As Long, lpANDbitPlane As Any, lpXORbitPlane As Any) As Long
```

パラメータ

- *hinst*　マウスカーソルを作成するアプリケーションの現在のインスタンスを識別する。
- *xHotSpot*　マウスカーソルのホットスポットの水平位置を指定する。
- *yHotSpot*　マウスカーソルのホットスポットの垂直位置を指定する。
- *nWidth*　マウスカーソルの幅をピクセル単位で指定する。
- *nHeight*　マウスカーソルの高さをピクセル単位で指定する。
- *pvANDplane*　マウスカーソルのANDマスク用のビット値を表すバイト配列を指すポインタ。これはデバイスに依存するモノクロビットマップでのビットマスク。
- *pvXORplane*　マウスカーソルのXORマスク用のビット値を表すバイト配列を指すポインタ。これはデバイスに依存するモノクロビットマップでのビットマスク。

戻り値

正常終了	マウスカーソルのハンドル
異常終了	NULL

拡張エラー情報を取得するには、GetLastError関数を利用する。

解　説

指定されたサイズ、ビットパターン、およびホットスポットを持つマウスカーソルを作成する。*nWidth*パラメータと*nHeight*パラメータには、現在のディスプレイドライバがサポートする幅と高さを指定しなければならない。これはシステムがほかのサイズのマウスカーソルを作成できないためである。

参　照

CreateIcon,DestroyCursor,GetModuleHandle,→GetSystemMetrics,SetCursor

GetCursorPos	user32.dll	95	NT
カーソルの現在のスクリーン座標の取得			

V　C

```
BOOL GetCursorPos (
    LPPOINT    lpPoint    //POINT構造体へのポインタ
    );
```

V　B

Declare Function GetCursorPos Lib "user32" Alias "GetCursorPos" (lpPoint As POINTAPI) As Long

パラメータ

・*lpPoint*　カーソル位置をスクリーン座標で受け取るPOINT構造体へのポインタである。

戻り値

正常終了	TRUE
異常終了	FALSE

拡張エラー情報を取得するには、GetLastError関数を利用する。

解　説

カーソルの現在の位置のスクリーン座標を取得する。
カーソルの位置は常にスクリーン座標で与えられる。カーソルが表示されているウィンドウのマッピングモードには関係がない。呼び出し側プロセスは、ウィンドウステーションへのWINSTA_READATTRIBUTESアクセスを持っていなければならない。

参　照

ClipCursor,SetCursor,SetCursorPos,ShowCursor,→POINT

LoadCursor	user32.dll	95	NT
リソースからマウスカーソルをロード			

V　C

```
HCURSOR LoadCursor(
    HINSTANCE    hInstance,      //アプリケーションのインスタンスを識別する
                                   ハンドル
    LPCTSTR      lpCursorName //カーソルリソースの識別子
    );
```

V　B

Declare Function LoadCursor Lib "user32" Alias "LoadCursorA" (ByVal hInstance As Long, ByVal lpCursorName As String) As Long

パラメータ

・*hInstance*　ロードするカーソルリソースを含む実行可能ファイルのインスタンスを識別するハンドルを指定する。

・*lpCursorName*　ロードするカーソルリソースを識別するNULLで終わる文字列へのポインタ。Win32ではあらかじめ予約された次の値を指定することもできる。

値	意味
IDC_APPSTARTING	標準の矢印と小さな砂時計
IDC_ARROW	標準の矢印
IDC_CROSS	クロスヘアカーソル
IDC_IBEAM	Iビームカーソル
IDC_ICON	WindowsNT：Emptyアイコン
IDC_NO	スラッシュ付き円
IDC_SIZE	WindowsNT：4方向矢印
IDC_SIZEALL	4方向矢印
IDC_SIZENESW	右上・左下両向き矢印
IDC_SIZENS	上下両向き矢印
IDC_SIZENWSE	左上・右下両向き矢印
IDC_SIZEWE	左右両向き矢印
IDC_UPARROW	上向き矢印
IDC_WAIT	砂時計

戻り値

正常終了	ロードされたカーソルのハンドル
異常終了	NULL

拡張エラー情報はGetLastError関数で取得する。

解　説

*lpCursorName*がカーソル（アイコンなど）を除いたほかのリソースタイプを指定しても、戻り値はNULLとならない。LoadCursor機能は、現在のディスプレイに最適なカーソルリソースを捜す。カーソルはカラーまたはモノクロのビットマップである。

参　照

→LoadImage,→MAKEINTRESOURCE,SetCursor,SetCursorPos,→ShowCursor

LoadCursorFromFile

user32.dll | 95 | NT

ファイルからカーソルをロード

V C

```
HCURSOR LoadCursorFromFile (
    LPCTSTR  lpFileName    //カーソルファイルもしくはシステムカーソル
    );
```

V B

```
Declare Function LoadCursorFromFile Lib "user32" Alias "LoadCursorFromFileA"
(ByVal lpFileName As String) As Long
```

パラメータ

・*lpFileName*　.CURもしくは.ANIの拡張子で表されるカーソルファイルを指定するNULLで終わる文字列へのポインタを指定する。上位ワードが0で、下位ワードにシステムカーソルを識別する値を指定することも可能。

システムカーソル名	識別子
"Arrow"	OCR_NORMAL
"IBeam"	OCR_IBEAM
"Wait"	OCR_WAIT
"Crosshair"	OCR_CROSS
"UpArrow"	OCR_UP
"Size"	OCR_SIZE
"Icon"	OCR_ICON
"SizeNWSE"	OCR_SIZENWSE
"SizeNESW"	OCR_SIZENESW
"SizeWE"	OCR_SIZEWE
"SizeNS"	OCR_SIZENS
"SizeAll"	OCR_SIZEALL
"No"	OCR_NO
"AppStarting"	OCR_APPSTARTING

たとえば、LoadCursorFromFile((LPWSTR)OCR_NORMAL)というように呼び出す。

戻り値

正常終了	ロードされたカーソルのハンドル
異常終了	NULL

拡張エラー情報はGetLastError関数で取得する。GetLastError関数は次の値を返す。

値	意味
ERROR_FILE_NOT_FOUND	指定されたファイルが見つからない。

参　照

→LoadCursor,SetCursor,SetSystemCursor

ShowCursor	user32.dll	95	NT
マウスカーソル表示カウントの増減			

V C

```
int ShowCursor(
    BOOL        bShow     //カーソル表示フラグ
    );
```

V B

Declare Function ShowCursor Lib "user32" Alias "ShowCursor" (ByVal bShow As Long) As Long

パラメータ

・*bShow*　表示カウンターを増加させるか減少させるかを指定する。*bShow*がTRUEの場合には増加、FALSEの場合には減少される。

戻り値

新しい表示カウント。

解　説

ShowCursor関数はカーソルの表示、非表示を決定する内部の表示カウンターをセットする。 表示カウントが0以上の場合、カーソルは表示される。初期の表示カウントは0である。マウスがインストールされていない場合は表示カウントは-1となる。

参　照

ClipCursor,→GetCursorPos,SetCursor,SetCursorPos

Icon

ArrangeIconicWindows	user32.dll	95	NT
アイコンを整列			

V C

```
UINT ArrangeIconicWindows (
    HWND        hwnd      //親ウィンドウハンドル
    );
```

V B

Declare Function ArrangeIconicWindows Lib "user32" Alias "ArrangeIconicWindows" (ByVal hwnd As Long) As Long

パラメータ

・*hwnd*　親ウィンドウハンドル

戻り値

正常終了	アイコン1列分の高さ
異常終了	0

拡張エラー情報を取得するには、GetLastError関数を利用する。

解　説

親ウィンドウ中のアイコン化されたすべてのウィンドウを整列する。
デスクトップウィンドウのハンドルはGetDesktopWindow関数を使用して取得する。

参　照

GetDesktopWindow

CreateIconFromResource	user32.dll	95	NT
リソースからアイコンリソースをロード			

VC

```
HICON CreateIconFromResource(
        PBYTE    presbits,     //アイコンもしくはカーソルビットへのポインタ
        DWORD    dwResSize,    //ビットバッファのバイト数
        BOOL     fIcon,        //アイコンフラグ
        DWORD    dwVer         //Windowsフォーマットバージョン
        );
```

VB

Declare Function CreateIconFromResource Lib "user32" Alias "CreateIconFromResource" (presbits As Byte, ByVal dwResSize As Long, ByVal fIcon As Long, ByVal dwVer As Long) As Long

パラメータ

・*presbits*　アイコンまたはカーソルリソースビットを含むバッファへのポインタ。これらのビットは関数の呼び出しにより、LookupIconIdFromDirectory（Windows 95では LookupIconIdFromDirectoryExでも可能）とLoadResource関数でロードされる。

・*dwResSize*　*presbits*パラメータで示されたビットセットのサイズをバイト単位で指定する。

・*fIcon*　アイコンまたはカーソルが作成する必要があるかどうかを指定する。このパラメータがTRUEの場合には、アイコンが作成される。FALSEの場合にはカーソルが作成される。

・*dwVer*　*presbits*パラメータで示されたリソースビットのためのアイコンのバージョン番号またはカーソル形式を指定する。このパラメーターは次の値のうちの1つである。

フォーマット	値
Windows 2.x	0x00020000
Windows 3.x	0x00030000

すべてのWin32アプリケーションで使用するアイコンおよびカーソルは、Windows 3.x互換形式である。

戻り値

正常終了	アイコンもしくはカーソルのハンドル
異常終了	NULL

拡張エラー情報はGetLastError関数で取得する。

参照

CreateIconFromResource,CreateIconFromResourceEx,CreateIconIndirect,GetIconInfo,
→LoadResource,LookupIconIdFromDirectory,LookupIconIdFromDirectoryEx

Message and Message Queue

DispatchMessage	user32.dll	95	NT
メッセージのディスパッチ			

V　C

```
LONG DispatchMessage (
    CONST MSG    *lpmsg    //メッセージを含むMSG構造体へのポインタ
    );
```

V　B

Declare Function DispatchMessage Lib "user32" Alias "DispatchMessageA" (lpMsg As MSG) As Long

パラメータ

・*lpmsg*　メッセージを含むMSG構造体へのポインタである。

戻り値

戻り値は、ウィンドウプロシージャが返す値を示す。戻り値の意味はディスパッチされたメッセージにより異なるが、一般に戻り値は無視される。

解　説

ウィンドウプロシージャにメッセージをディスパッチする。通常は、GetMessage関数で取得したメッセージをディスパッチするために利用される。
MSG構造体は、有効なメッセージの値を含んでいなければならない。*lpmsg*パラメータがWM_TIMERメッセージを指すポインタで、WM_TIMERメッセージの*lParam*パラメータがNULLでない場合、*lParam*はウィンドウプロシージャの代わりに呼び出される関数を指すポインタとなる。

参　照

GetMessage,PeekMessage,PostAppMessage,PostMessage,TranslateMessage,→MSG,→WM_TIMER

GetMessage	user32.dll	95	NT
メッセージキューからメッセージを取得			

V C

```
BOOL GetMessage(
    LPMSG   lpMsg,          //MSG構造体へのポインタ
    HWND    hWnd,           //ウィンドウを識別するハンドル
    UINT    wMsgFilterMin,  //最初のメッセージ識別子
    UINT    wMsgFilterMax   //最後のメッセージ識別子
    );
```

V B

Declare Function GetMessage Lib "user32" Alias "GetMessageA" (lpMsg As MSG, ByVal hwnd As Long, ByVal wMsgFilterMin As Long, ByVal wMsgFilterMax As Long) As Long

パラメータ

・*lpMsg* メッセージキューから取り出すメッセージを識別するMSG構造体へのポインタを指定する。

・*hWnd* メッセージを取り出すウィンドウを識別するハンドルを指定する。次の値のいずれかはそれぞれ予約された特殊な意味を持つ。

値	意味
NULL	GetMessageは、PostThreadMessageを経たあらゆるウィンドウメッセージも取り出す。
wMsgFilterMin	取り出される最も低いメッセージ値の整数値を指定する。
wMsgFilterMax	取り出される最も高いメッセージ値の整数値を指定する。メッセージを含むMSG構造体へのポインタである。

戻り値

メッセージを取得した場合	TRUE
WM_QUITを取得した場合	FALSE
異常終了	-1

解　説

メッセージループを終えプログラムを終了するかを決めるために、アプリケーションが戻り値を用いる。

参　照

IsChild,→MSG,PeekMessage,PostMessage,PostThreadMessage,WaitMessage

SendMessage	user32.dll	95	NT
メッセージの送信			

V C

```
LRESULT SendMessage(
        HWND        hWnd,     //送信先ウィンドウを識別するハンドル
        UINT        Msg,      //送信するメッセージ
        WPARAM      wParam,   //メッセージ第1パラメータ
        LPARAM      lParam    //メッセージ第2パラメータ
        );
```

V B

```
Declare Function SendMessage Lib "user32" Alias "SendMessageA" (ByVal hwnd As Long, ByVal wMsg As Long, ByVal wParam As Long, lParam As Long) As Long
```

パラメータ

- *hWnd*　Windowsがメッセージを受け取るウィンドウを識別する。パラメータがHWND_BROADCASTの場合は、メッセージはシステムのすべてのトップレベルウィンドウに送られるが、子ウィンドウには送信されない。
- *Msg*　送信されるメッセージを指定する。
- *wParam*　メッセージの第1パラメータを指定する。
- *lParam*　メッセージキューから取り出すメッセージを識別するMSG構造体へのポインタを指定する。
- *hWnd*　メッセージを取り出すウィンドウのハンドルを指定する。次の値は、いずれも予約された特殊な意味を持つ。

値	意味
NULL	GetMessageは、PostThreadMessageを経たあらゆるウィンドウメッセージも取り出す。
wMsgFilterMin	取り出される最も低いメッセージ値の整数値を指定する。
wMsgFilterMax	取り出される最も高いメッセージ値の整数値を指定する。メッセージを含むMSG構造体へのポインタである。

戻り値

送信されたメッセージにより異なる。

解　説

指定されたウィンドウが呼び出し元スレッドにより作成されている場合は、Windowsが直ちにサブルーチンとして呼ばれる。指定されたウィンドウが、異なるスレッドで作成されている場合には、Windowsはそのスレッドに処理を移す。

参　照

InSendMessage,PostMessage,SendDlgItemMessage

Mouse Input

ReleaseCapture	user32.dll	95	NT
マウスキャプチャーを終了する			

V C BOOL ReleaseCapture();

V B Declare Function ReleaseCapture Lib "user32" Alias "ReleaseCapture" () As Long

戻り値

正常終了	TRUE
異常終了	FALSE

解 説 アプリケーションは、SetCapture関数を呼んだ後にこの関数を呼ぶ。

参 照 GetCapture,→SetCapture,→WM_CAPTURECHANGED

SetCapture	user32.dll	95	NT
マウスキャプチャーを開始する			

V C

```
HWND SetCapture(
    HWND        hWnd    //ウィンドウを識別するハンドル
    );
```

V B Declare Function SetCapture Lib "user32" Alias "SetCapture" (ByVal hwnd As Long) As Long

パラメータ

・*hwnd* マウスキャプチャーを開始するウィンドウを識別するためのハンドルを指定する。

戻り値

正常終了	以前のマウスキャプチャーが開始されていたウィンドウハンドル
異常終了	NULL

解 説 アクティブウィンドウだけがマウスキャプチャーを実行できる。

参 照 GetCapture,→ReleaseCapture,→WM_CAPTURECHANGED

SetDoubleClickTime	user32.dll	95	NT
ダブルクリックの間隔を設定する			

V C

```
BOOL SetDoubleClickTime(
    UINT    uInterval        //ダブルクリック間隔
    );
```

V B

```
Declare Function SetDoubleClickTime Lib "user32" Alias "SetDoubleClickTime"
(ByVal wCount As Long) As Long
```

パラメータ

・*uInterval*　ダブルクリックの間隔をミリ秒単位で指定する。パラメータが0に設定された場合には、デフォルトであるダブルクリック間隔が500ミリ秒となる。

戻り値

正常終了	TRUE
異常終了	FALSE

解　説

SetDoubleClickTime関数はシステムのすべてのウィンドウのダブルクリック間隔を変更する。

参　照

GetDoubleClickTime

Painting and Drawing

BeginPaint	user32.dll	95	NT
描画準備を行う			

V C

```
HDC BeginPaint (
    HWND            hwnd,  //再描画するウィンドウハンドル
    LPPAINTSTRUCT   lpps,  //描画情報を受け取るPAINTSTRUCT構造体への
                             ポインタ
    );
```

V B

```
Declare Function BeginPaint Lib "user32" Alias "BeginPaint" (ByVal hwnd As Long,
lpPaint As PAINTSTRUCT) As Long
```

パラメータ

・*hwnd*　再描画するウィンドウを識別する。

・*lpps*　描画情報を受け取るPAINTSTRUCT構造体へのポインタである。

戻り値

正常終了	ディスプレイデバイスコンテキスト（DC）のハンドル
異常終了	NULL

解　説

指定されたウィンドウに対して描画の準備をし、PAINTSTRUCT構造体に描画に関する情報を格納する。

描画される領域にキャレットがある場合、BeginPaint関数は消去されないように自動的にキャレットを隠す。ウィンドウのクラスが背景色のブラシを持っている場合、BeginPaint関数が制御を戻す前にそのブラシを使用して、更新リージョンの背景を消去する。

参　照

EndPaint,InvalidateRect,InvalidateRgn,ValidateRect,ValidateRgn,→PAINTSTRUCT, →RECT,→WM_PAINT,→WM_ERASEBKGND

DrawAnimatedRects	user32.dll	95	NT
アニメーション描画領域の作成			

V　C

```
BOOL WINAPI DrawAnimatedRects(
        HWND            hwnd,       //ウィンドウを識別するハンドル
        int             idAni,      //アニメーションハンドル
        CONST RECT      *lprcFrom,//SMALL_RECT構造体へのポインタ
        CONST RECT      *lprcTo     //SMALL_RECT構造体へのポインタ
        );
```

V　B

Declare Function DrawAnimatedRects Lib "user32" Alias "DrawAnimatedRects" (ByVal hwnd As Long, ByVal idAni As Long, lprcFrom As Rect, lprcTo As Rect) As Boolean

パラメータ

- *hwnd*　長方形領域をクリップされたウィンドウを識別するハンドルを指定する。
- *idAni*　0を指定しなければならない。
- *lprcFrom*　最小化されたウィンドウの位置とサイズを指定するSMALL_RECT構造体へのポインタを指定する。
- *lprcTo*　元の大きさに戻されたウィンドウの位置とサイズを指定するSMALL_RECT構造体へのポインタを指定する。

戻り値

正常終了	TRUE
異常終了	FALSE

参　照

→SMALL_RECT

RedrawWindow	user32.dll	95	NT
ウィンドウを再描画する			

V　C

```
BOOL RedrawWindow(
    HWND          hWnd,          //ウィンドウを識別するハンドル
    CONST RECT    *lprcUpdate,   //RECT構造体へのポインタ
    HRGN          hrgnUpdate,    //更新するリージョンを識別するハンドル
    UINT          flags          //再描画フラグの配列
    );
```

V　B

```
Declare Function RedrawWindow Lib "user32" Alias "RedrawWindow" (ByVal hwnd As Long, lprcUpdate As RECT, ByVal hrgnUpdate As Long, ByVal fuRedraw As Long) As Long
```

パラメータ

- *hwnd*　再描画するウィンドウを識別するハンドルを指定する。NULLが指定された場合は、デスクトップが再描画される。
- *lprcUpdate*　再描画する長方形領域を指定するRECT構造体へのポインタを指定する。*hrgnUpdate*パラメータが領域を識別する場合、このパラメータは無視される。
- *hrgnUpdate*　更新領域を識別する。*hrgnUpdate*と*lprcUpdate*パラメータの両方がNULLの場合、クライアント領域全体が更新領域に指定される。
- *flags*　1つ以上の再描画フラグを指定する。
  ウィンドウもしくはコントロールの再描画を無効にするために、次の値のいずれかを指定する。

値（無効化）	意味
RDW_ERASE	ウィンドウの再描画時にWM_ERASEBKGNDメッセージを受け取る。RDW_INVALIDATEフラグを同時に指定しなければならない。
RDW_FRAME	WM_NCPAINTメッセージを受け取る。RDW_INVALIDATEフラグを同時に指定しなければならない。RDW_UPDATENOWまたはRDW_ERASENOWが指定されない限り、WM_NCPAINTメッセージはRedrawWindowの実行の間に送られない。
RDW_INTERNALPAINT	無効部分に関係なくWM_PAINTメッセージを送信する。
RDW_INVALIDATE	*lprcUpdate*または*hrgnUpdate*を無効にする。両パラメータがNULLの場合には、ウィンドウ全体が無効化される。

ウィンドウもしくはコントロールの再描画を有効にするために、次の値のいずれかを指定する。

値（有効化）	意味
RDW_NOERASE	いかなるWM_ERASEBKGNDメッセージも抑制する
RDW_NOFRAME	いかなるWM_NCPAINTメッセージも抑制する。RDW_VALIDATEと同時に指定しなければならない。
RDW_NOINTERNALPAINT	いかなる内部WM_PAINTメッセージも抑制する。NULLでない更新領域には一切影響しない。
RDW_VALIDATE	*lprcUpdate*または*hrgnUpdate*を有効にする両パラメータがNULLの場合には、ウィンドウ全体が有効になる。内部のWM_PAINTメッセージには影響しない。

再描画を実行する場合には、次のフラグのいずれかを必ず指定しなければならない。

値	意味
RDW_ERASENOW	対象となるウィンドウは制御が戻る前にWM_NCPAINTとWM_ERASEBKGNDメッセージを受け取る。WM_PAINTメッセージは通常のスケジュールで受け取られる。
RDW_UPDATENOW	対象となるウィンドウは制御が戻る前にWM_NCPAINT、WM_ERASEBKGND、およびWM_PAINTメッセージを受け取る。

デフォルトでは、RedrawWindowにより影響を受けるウィンドウはWS_CLIPCHILDRENスタイルを持っているかどうかに依存している。ウィンドウがRedrawWindow関数により影響を受けるかどうかは次のフラグで指定する。

値	意味
RDW_ALLCHILDREN	子ウィンドウが存在する場合に再描画の対象とする。
RDW_NOCHILDREN	子ウィンドウが存在する場合に再描画の対象としない再描画するウィンドウを識別する。

・*lpps*　　描画情報を受け取るPAINTSTRUCT構造体へのポインタである。

戻り値

正常終了	TRUE
異常終了	FALSE

拡張エラー情報はGetLastError関数で取得する。

解　説

デスクトップウィンドウの一部を無効にするためにRedrawWindowを使用する場合、デスクトップウィンドウはWM_PAINTメッセージを受け取らない。デスクトップを再描画するためにWM_ERASEBKGNDメッセージを生成させる場合、アプリケーションはRDW_ERASEフラグを使う。

参　照

→GetUpdateRect,GetUpdateRgn,InvalidateRect,InvalidateRgn,→RECT,
→UpdateWindow

UpdateWindow	user32.dll	95	NT
ウィンドウを更新する			

V　C

```
BOOL UpdateWindow(
    HWND        hWnd      //ウィンドウを識別するハンドル
    );
```

V　B

Declare Function UpdateWindow Lib "user32" Alias "UpdateWindow" (ByVal hwnd As Long) As Long

パラメータ

・*hwnd*　　再描画するウィンドウを識別するハンドルを指定する。

戻り値

正常終了	TRUE
異常終了	FALSE

参　照

ExcludeUpdateRgn,→GetUpdateRect,→GetUpdateRgn,InvalidateRect,InvalidateRgn,→WM_PAINT

Rectangle

IntersectRect	user32.dll	95	NT
2つの長方形の重なりを計算			

V　C

```
BOOL IntersectRect(
    LPRECT      lprcDst,    //重なりの結果を示すRECT構造体へのポインタ
    CONST RECT  *lprcSrc1,  //第1の長方形を示すRECT構造体へのポインタ
    CONST RECT  *lprcSrc2   //第2の長方形を示すRECT構造体へのポインタ
    );
```

V　B

Declare Function IntersectRect Lib "user32" Alias "IntersectRect" (lpDestRect As RECT, lpSrc1Rect As RECT, lpSrc2Rect As RECT) As Long

パラメータ

・*lprcDst*　　2つの長方形の重なりを受け取るためのRECT構造体へのポインタを指定する。

・*lprcSrc1*　　第1の長方形を指定するRECT構造体へのポインタを指定する。

・*lprcSrc2*　　第2の長方形を指定するRECT構造体へのポインタを指定する。

戻り値

重なりがある場合	TRUE
それ以外	FALSE

拡張エラー情報はGetLastError関数で取得する。

参　照

InflateRect,→OffsetRect,→RECT,UnionRect

OffsetRect	user32.dll	95	NT
指定された長方形を移動する			

V　C

```
BOOL OffsetRect(
    LPRECT  lprc,   //RECT構造体へのポインタ
    int     dx,     //水平移動位置
    int     dy      //垂直移動位置
    );
```

V　B

Declare Function OffsetRect Lib "user32" Alias "OffsetRect" (lpRect As RECT, ByVal x As Long, ByVal y As Long) As Long

パラメータ

- *lprc*　移動する長方形の論理座標を格納するRECT構造体へのポインタを指定する。
- *dx*　長方形を水平移動するための値を指定する。左方向は負の値、右方向は正の値を指定する。
- *dy*　長方形を垂直移動するための値を指定する。上方向は負の値、下方向は正の値を指定する。

戻り値

正常終了	TRUE
異常終了	FALSE

拡張エラー情報はGetLastError関数で取得する。

参　照

InflateRect,→IntersectRect,UnionRect,→RECT

SetRect	user32.dll	95	NT
長方形の座標領域を設定			

V　C

```
BOOL SetRect(
    LPRECT  lprc,      //RECT構造体へのポインタ
    int     xLeft,     //左端のX座標
    int     yTop,      //上端のY座標
    int     xRight,    //右端のX座標
    int     yBottom    //下端のY座標
    );
```

V　B

Declare Function SetRect Lib "user32" Alias "SetSelect" (lpRect As Rect, ByVal X1 As Long, ByVal Y1 As Long, ByVal X2 As Long, ByVal Y2 As Long) As Long

パラメータ

- *lprc*　設定された長方形の論理座標を受け取るRECT構造体へのポインタを指定する。
- *xLeft*　設定される長方形の左端のX座標を指定する。
- *yTop*　設定される長方形の上端のY座標を指定する。
- *xRight*　設定される長方形の右端のX座標を指定する。
- *yBottom*　設定される長方形の下端のY座標を指定する。

戻り値

正常終了	TRUE
異常終了	FALSE

拡張エラー情報はGetLastError関数で取得する。

参照

CopyRect,SetRectEmpty,→RECT

Timers

KillTimer	user32.dll	95	NT
タイマーイベントを削除			

VC

```
BOOL KillTimer(
        HWND          hWnd,      //ウィンドウを識別するハンドル
        UINT          uIDEvent   //タイマー識別子
        );
```

VB

```
Declare Function KillTimer Lib "user32" Alias "KillTimer" (ByVal hwnd As Long,
ByVal nIDEvent As Long) As Long
```

パラメータ

- *hWnd*　削除するタイマーと関連付けられたウィンドウを識別するハンドルを指定する。
- *uIDEvent*　削除されるタイマーを指定する。*hWnd*にNULLを指定しSetTimer関数を呼び出した場合、このパラメータは、SetTimerにより戻されたタイマー識別子である。

戻り値

正常終了	TRUE
異常終了	FALSE

拡張エラー情報はGetLastError関数で取得する。

参照

→SetTimer,→WM_TIMER

SetTimer	user32.dll	95	NT
システムタイマーを作成			

V C

```
UINT SetTimer(
    HWND        hWnd,          //ウィンドウを識別するハンドル
    UINT        nIDEvent,      //タイマー識別子
    UINT        uElapse,       //タイマーイベント間隔
    TIMERPROC   lpTimerFunc    //タイマーイベントプロシージャへのポインタ
    );
```

V B

```
Declare Function SetTimer Lib "user32" Alias "SetTimer" (ByVal hWnd As Long, ByVal nIDEvent As Long, ByVal uElapse As Long,, ByVal lpTimerFunc As Long) As Long
```

パラメータ

- *hWnd* タイマーイベントと関連付けられたウィンドウを識別するハンドルを指定する。パラメータにNULLを指定した場合、ウィンドウはタイマーと結び付けられず、*nIDEvent*パラメータは無視される。
- *nIDEvent* 0以外のタイマー識別子を指定する。*hWnd*パラメータがNULLの場合、このパラメータは無視される。
- *uElapse* タイマーイベント間隔をミリ秒で指定する。
- *lpTimerFunc* タイマーイベントが発生した際に実行される関数へのポインタを指定する。*lpTimerFunc*パラメータがNULLの場合、システムはWM_TIMERメッセージを送信する。MSG構造体のhwndメンバは*hWnd*パラメータの値を含む。

戻り値

正常終了	タイマー識別子
異常終了	0

参照

→KillTimer,→MSG,TimerProc,→WM_TIMER

Window Class

GetClassInfo

user32.dll | 95 | NT

クラス情報の取得

VC

```
BOOL GetClassInfo(
    HINSTANCE    hInstance,     //アプリケーションのインスタンスを識別するハンドル
    LPCTSTR      lpClassName,   //クラス名を格納する文字列へのポインタ
    LPWNDCLASS   lpWndClass     //WNDCLASS構造体へのポインタ
    );
```

VB

Declare Function GetClassInfo Lib "user32" Alias "GetClassInfoA" (ByVal hInstance As Long, ByVal lpClassName As String, lpWndClass As WNDCLASS) As Long

パラメータ

- *hInstance*　クラスを作成したアプリケーションを識別するハンドルを指定する。
- *lpClassName*　クラス名を含むNULLで終わる文字列へのポインタを指定する。
- *lpWndClass*　クラス情報を格納するWNDCLASS構造体へのポインタを指定する。

戻り値

正常終了	TRUE
異常終了	FALSE

拡張エラー情報はGetLastError関数で取得する。

参照

GetClassLong,GetClassName,GetClassWord,GlobalAddAtom,→RegisterClass, →WNDCLASS

RegisterClass

user32.dll | 95 | NT

クラス情報を登録

VC

```
ATOM RegisterClass(
    CONST WNDCLASS  *lpWndClass //WNDCLASS構造体へのポインタ
    );
```

VB

Declare Function RegisterClass Lib "user32" Alias "RegisterClass" (Class As WNDCLASS) As Long

パラメータ

- *lpWndClass*　登録するクラス情報を格納したWNDCLASS構造体へのポインタを指定する。

戻り値

正常終了	登録されたクラスを識別するアトム
異常終了	0

拡張エラー情報はGetLastError関数で取得する。

解　説

登録されたすべてのクラスは、アプリケーション終了時にすべての登録を抹消される。

参　照

→CreateWindow,CreateWindowEx,→GetClassInfo,GetClassName,→UnregisterClass,→WindowProc,→WNDCLASS

UnregisterClass	user32.dll	95	NT
クラス情報をクラステーブルから削除			

V　C

```
BOOL UnregisterClass(
    LPCTSTR     lpClassName,  //クラス名を格納したNULLで終わる文字列へのポインタ
    HINSTANCE   hInstance     //アプリケーションのインスタンスを識別するハンドル
    );
```

V　B

Declare Function UnregisterClass Lib "user32" Alias "UnregisterClassA" (ByVal lpClassName As String, ByVal hInstance As Long) As Long

パラメータ

- *lpClassName*　クラス名を格納するNULLで終わる文字列へのポインタを指定する。
- *hInstance*　クラスを作成したアプリケーションのインスタンスを識別するハンドルを指定する。登録するクラス情報を格納したWNDCLASS構造体へのポインタを指定する。

戻り値

正常終了	TRUE
異常終了	FALSE

拡張エラー情報はGetLastError関数で取得する。

解　説

関数を実行する前に、アプリケーションは削除するクラスにより作成されたすべてのウィンドウを破棄しなければならない。

参　照

GlobalAddAtom,→RegisterClass

Window Property

GetProp	user32.dll	95	NT
プロパティリストの取得			

V C

```
HANDLE GetProp(
    HWND        hWnd,      //ウィンドウを識別するハンドル
    LPCTSTR     lpString   //文字列もしくはアトム
);
```

V B

Declare Function GetProp Lib "user32" Alias "GetPropA" (ByVal hwnd As Long, ByVal lpString As String) As Long

パラメータ

- *hWnd*　プロパティリストを検索するウィンドウを識別するハンドルを指定する。
- *lpString*　NULLで終わる文字列へのポインタもしくは文字列を識別するアトムを指定する。アトムを指定する場合、GlobalAddAtom関数で得た値でなければばならない。アトム（16ビット）は、*lpString*パラメータの下位ワードに置かれ、上位ワードは0である。

戻り値

プロパティリストに文字列を発見した場合	ハンドル
それ以外	NULL

参照

EnumProps,GlobalAddAtom,RemoveProp,→SetProp

SetProp	user32.dll	95	NT
プロパティリストの設定			

V C

```
BOOL SetProp(
    HWND        hWnd,       //ウィンドウを識別するハンドル
    LPCTSTR     lpString,   //文字列へのポインタもしくはアトム
    HANDLE      hData       //プロパティ値へのハンドル
);
```

V B

Declare Function SetProp Lib "user32" Alias "SetPropA" (ByVal hwnd As Long, ByVal lpString As String, ByVal hData As Long) As Long

パラメータ

- *hWnd*　エントリを追加するプロパティリストを含むウィンドウを識別するハンドルを指定する。

・*lpString*　NULLで終わる文字列へのポインタ、もしくは文字列を識別するアトムを指定する。アトムを指定する場合、GlobalAddAtom関数で得た値でなければならない。アトム（16ビット）は、*lpString*パラメータの下位ワードに置かれ、上位ワードは0である。

・*hData*　プロパティリストに格納するデータを識別するためのハンドルを指定する。

戻り値

正常終了	TRUE
異常終了	FALSE

参照

EnumProps,EnumPropsEx,→GetProp,GlobalAddAtom,RemoveProp

2. 3 Windows Controls
ウィンドウコントロール

Windowsアプリケーションでは、システムが提供する各種のコントロールオブジェクトを利用してプログラミングを進める。これらのコントロールを利用することで、作成されるアプリケーションは標準的かつ共通のユーザーインターフェイスオブジェクトを簡単に利用できる。

CommonControls

GetEffectiveClientRect	――――――――	95	NT
クライアント領域矩形範囲計算			

V　C

```
void GetEffectiveClientRect(
        HWND      hWnd,     //ウィンドウハンドル
        LPRECT    lprc,     //RECT構造体へのポインタ
        PINT      lpInfo    //コントロール識別子の配列へのポインタ
        );
```

パラメータ

- *hWnd*　対象となるクライアント領域のウィンドウハンドル。
- *lprc*　矩形範囲がセットされるRECT構造体へのポインタ。
- *lpInfo*　クライアント領域上のコントロールの16ビットコントロール識別子の配列へのポインタ。各コントロールは2つの配列要素からなる。1番目の要素は、0以外でなければならない。2番目の要素が、コントロール識別子である。この配列の最後の要素は0でなければならない。

解　説

GetEffectiveClientRect関数は、クライアント領域の矩形範囲を計算する。この関数はアプリケーションを制限する。

参　照

→RECT

InitCommonControls

		95	NT

コモンコントロールDLL確立

V C

```
void InitCommonControls();
```

解 説

InitCommonControls関数は、ロードされたコモンコントロールダイナミックリンクライブラリ(DLL)を確立する。

ShowHideMenuCtl

		95	NT

メニュー項目チェック属性と表示非表示設定

V C

```
BOOL ShowHideMenuCtl(
    HWND    hWnd,    //MDI子ウィンドウハンドル
    UINT    uFlags,  //メッセージ
    LPINT   lpInfo   //最初のメッセージパラメータ
    );
```

パラメータ

- *hWnd* メニューやコントロールのあるウィンドウハンドル。
- *uFlags* チェックマークの受け渡しのメニュー項目識別子。
- *lpInfo* 値の組み合わせの入った配列のポインタ。最初の組み合わせの2番目の値はアプリケーションのメニュー項目ハンドルである。次にメニュー項目識別子とコントロールウィンドウ識別子の組み合わせが続く。この関数は、*uFlags*にあった値を検出する。値が検出されたら、メニュー項目をチェックまたはチェックを取り消し、対応するコントロールを表示または非表示にする。

戻り値

処理を実行	TRUE
処理を実行しなかった	FALSE

解 説

ShowHideMenuCtl関数は、指定されたメニュー項目のチェックマーク属性のセットおよび削除をする。また、対応するコントロールを表示や非表示にする。この関数は指定されたメニュー項目にチェックマークがない場合はセットし、対応するコントロールを表示する。メニュー項目にチェックマークがある場合は、チェックマークを取り消し、対応するコントロールを非表示にする。

DragListBox

DrawInsert	――――――	95	NT
アイコン描画			

V C

```
void DrawInsert(
        HWND    handParent,   //親ウィンドウハンドル
        HWND    hLB,          //ドラッグリストボックスハンドル
        int     nItem         //アイコン識別子
        );
```

パラメータ

・*handParent*　　ドラッグリストボックスの親ウィンドウハンドル。
・*hLB*　　ドラッグリストボックスハンドル。
・*nItem*　　描画されるアイコン識別子。

解　説

DrawInsert関数は、指定されたドラッグリストボックスの親ウィンドウに挿入アイコンを描画する。

LBItemFromPt	――――――	95	NT
リスト項目インデックス検出			

V C

```
int LBItemFromPt(
`       HWND    hLB,          //リストボックスハンドル
        POINT   pt,           //POINT構造体
        BOOL    bAutoScroll   //スクロールフラグ
        );
```

パラメータ

・*hLB*　　チェックするリストボックスハンドル。
・*pt*　　チェックされた画面座標がセットされるPOINT構造体。
・*bAutoScroll*　　スクロールフラグ。このパラメータがTRUEで、ポインタがリストボックスを上下する場合、この関数はリストボックスを1行単位でスクロールさせ、-1を返す。それ以外の場合は、リストボックスのスクロールはしない。

戻り値

ポインタがリスト項目	項目識別子
それ以外	-1

LBItemFromPt関数はリストボックスで指定された先の項目インデックスを検出する。

参　照

→DL_DRAGGING,→POINT,→WM_MOUSEMOVE

MakeDragList	――――――――	95	NT
ドラッグリストボックス作成			

V C

```
BOOL MakeDragList(
        HWND    hLB     //リストボックスハンドル
        );
```

パラメータ

・*hLB*　　単一選択リストボックスハンドル。

戻り値

正常終了	TRUE
異常終了	FALSE

解　説

MakeDragList関数は単一選択リストボックスをドラッグリストボックスに変更する。

Property Sheet

AddPropSheetPageProc	――――――――	95	NT
プロパティシートページ追加			

V C

```
BOOL CALLBACK AddPropSheetPageProc(
    HPROPSHEETPAGE    hpage,      //プロパティシートページハンドル
    LPARAM            lParam      //32ビット値
    );
```

パラメータ

・*hpage*　　プロパティシートページハンドル。

・*lParam*　　アプリケーション定義の32ビット値。

戻り値

正常終了	TRUE
異常終了	FALSE

解　説

AddPropSheetPageProc関数は、プロパティシート拡張でプロパティシートにページを追加するときに使うアプリケーション定義のコールバック関数を指定する。

参　照

→HPROPSHEETPAGE,→CreatePropertySheetPage

CreatePropertySheetPage	――――――――	95	NT
新規プロパティシートページ作成			

V　C

```
HPROPSHEETPAGE    CreatProperty(
    LPCPROPSHEETPAGE   lppsp    //PROPSHEETPAGE構造体へのポインタ
    );
```

パラメータ

・*lppsp*　　プロパティシートのページを定義する、PROPSHEETPAGE構造体へのポインタ。

戻り値

正常終了	プロパティシートのハンドル
異常終了	FALSE

解　説

CreatePropertySheetPage関数は、プロパティシートに新規ページを作成する。

アプリケーションは、新規ページを含むプロパティシートの作成時にこの関数を使用し、既存のプロパティシートに新規ページを追加するときにPSM_ADDPAGEメッセージを使用する。

Windows 95では、最大16,364個のウィンドウハンドルをサポートする。

参　照

→PropertySheet,→PROPSHEETPAGE,→PSM_ADDPAGE

DestroyPropertySheetPage	――――――――	95	NT
プロパティシートページ破棄			

V　C

```
BOOL DestroyPropertySheetPage(
    HPROPSHEETPAGE    hPSPage  //プロパティシートページハンドル
    );
```

パラメータ

・*hPSPage*　　削除するプロパティシートページハンドル。

戻り値

正常終了	TRUE
異常終了	FALSE

解　説

DestroyPropertySheetPage関数は、プロパティシートページを破棄する。アプリケーションは、PropertySheet関数で使用されていないページに対しこの関数を使用する必要がある。

参　照

→PropertySheet

ExtensionPropSheetPageProc	――――――――	95	NT
プロパティシート拡張			

V C

```
BOOL CALLBACK ExtensionPropSheetPageProc(
    LPVOID                  lpv,      //プロパティシート項目記述ポインタ
    LPFNADDPROPSHEETPAGE    lpfnAddPropSheetPageProc,
                                      //AddPropSheetPageProc関数のポインタ
    LPARAM                  lParam    //32ビット値
    );
```

パラメータ

- *lpv*　作成されたプロパティシート項目を記述するアプリケーション定義値のポインタ。NULLのセットも可能。
- *lpfnAddPropSheetPageProc*　AddPropSheetPageProc関数のポインタ。拡張ダイナミックリンクライブラリ(DLL)でプロパティシートにページを追加するときこの関数を使用する。
- *lParam*　アプリケーション定義の32ビット値。

戻り値

正常終了	TRUE
異常終了	FALSE

解　説

ExtensionPropSheetPageProc関数は、プロパティシートを作成したモジュールに属するAddProp-SheetPageProc関数のアドレスを受け取るアプリケーション定義のコールバック関数を指定する。プロパティシートの拡張はこの関数をエクスポートしなければならない。

参　照

→AddPropSheetPageProc

PropertySheet	――――――――	95	NT
プロパティシート表示			

V C

```
int PropertySheet(
        LPCPROPSHEETHEADER    lppsph    //PROPSHEETHEADER構造体へのポ
                                          インタ
        );
```

パラメータ

- *lppsph*　プロパティシートのフレームやページを定義する、PROPSHEETHEADER構造体へのポインタ。

戻り値

正常終了	正の値
異常終了	-1

戻り値

PROPSHEETHEADER構造体のdwFlags項目がPSH_MODELESSフラグを指定する場合は、成功時に、プロパティシートダイアログのウィンドウハンドルを返す。以下の戻り値は、特別な意味を持つ。

値	意味
ID_PSREBOOTSYSTEM	ページはプロパティシートにPSM_REBOOTSYSTEMメッセージを送る。コンピュータはユーザーの変更を実施するため再起動しなければならない。
ID_PSRESTARTWINDOWS	ページはプロパティシートにPSM_RESTARTWINDOWSメッセージを送る。Windowsはユーザーの変更を実施するため再起動しなければならない。

解説

デフォルトでは、プロパティシートはモーダルダイアログを作成する。PROPSHEETHEADER構造体のdwFlags項目がPSH_MODELESSフラグを指定した場合は、プロパティシートは、モードレスダイアログを作成し、ダイアログ作成後すぐに制御を戻す。この場合、プロパティシートの戻り値は、モードレスダイアログのウィンドウハンドルである。
モードレスプロパティシートのため、メッセージループでプロパティシートダイアログメッセージを渡すためにPSM_ISDIALOGMESSAGEを使う。メッセージループでダイアログを破棄するとき終了するためにPSM_GETCURRENTPAGEHWNDを使う。ユーザーがOKかキャンセルボタンを選択したとき、PSM_GETCURRENTPAGEHWNDはNULLを返し、DestroyWindow関数によりダイアログを破棄する。

参照

→DestroyWindow,→PROPSHEETHEADER,→PSM_GETCURRENTPAGEHWND,
→PSM_ISDIALOGMESSAGE,→PSM_REBOOTSYSTEM,
→PSM_RESTARTWINDOWS

PropSheetPageProc | ―――――― | 95 | NT

プロパティシートページ初期化

VC

```
UINT CALLBACK PropSheetPageProc(
    HWND            hwnd,   //予備
    UINT            uMsg,   //アクションフラグ
    LPPROPSHEETPAGE ppsp    //PROPSHEETPAGE構造体へのポインタ
    );
```

パラメータ

- *hwnd*　将来のために予約。NULLでなければならない。
- *uMsg*　アクションフラグ。このパラメータは、以下の値である。

値	意味
PSPCB_CREATE	ページが作成される。ページの作成が許可された場合、0以外が返る。0の場合、作成されない。
PSPCB_RELEASE	ページが消去される。返り値は、無視される。

・*ppsp*　　ページの作成や消去の定義をするPROPSHEETPAGE構造体のポインタ。

戻り値

戻り値は、*uMsg*の値による。

解　説

PropSheetPageProc関数は、ページを作成したり破棄するときに、プロパティシートが呼ぶアプリケーション定義のコールバック関数を指定する。アプリケーションは、この関数を使いページの初期化やオペレーションをクリアする。アプリケーションは、CreatePropertySheetPage関数を呼びPROPSHEETPAGE構造体のアドレスを指定する前に、構造体のpfnCallback項目にコールバック関数のアドレスを指定しなければならない。

参　照

→CreatePropertySheetPage,→PROPSHEETPAGE

PropSheetProc	————	95	NT
プロパティシート初期化			

V　C

```
int CALLBACK PropSheetProc(
        HWND      hwndDlg,  //プロパティシートダイアログボックスのハンドル
        UINT      uMsg,     //アクションフラグ
        LPARAM    lParam    //0固定
        );
```

パラメータ

・*hwndDlg*　　プロパティシートダイアログボックスのハンドル。
・*uMsg*　　アクションフラグ。プロパティシートの初期化を示すPSCB_INITIALIZED値が定義される。
・*lParam*　　0固定。

戻り値

この関数は、0を返す。

解　説

関数は、プロパティシートの初期化のためシステムが呼ぶアプリケーション定義のコールバック関数を指定する。アプリケーションは、PropertySheet関数を呼びPROPSHEETHEADER構造体のアドレスを指定する前に、構造体のpfnCallback項目にコールバック関数のアドレスを指定しなければならない。

参　照

→PropertySheet,→PROPSHEETHEADER

Window

CreateStatusWindow	――――――	95	NT
ステータスウィンドウ作成			

V　C

```
HWND CreateStatusWindow(
    LONG      style,       //ウィンドウスタイル
    LPCTSTR   lpszText,    //ステータステキストポインタ
    HWND      hwndParent,  //親ウィンドウハンドル
    UINT      wID          //ステータスウィンドウのコントロール識別子
    );
```

パラメータ

- *style*　ステータスウィンドウのウィンドウスタイル。このパラメータにはWS_CHILDスタイルやWS_VISIBLEスタイルが含まれる。
- *lpszText*　最初の部分のステータステキストのNULLで終わる文字列のポインタ。
- *hwndParent*　親ウィンドウのハンドル。
- *wID*　ステータスウィンドウのコントロール識別子。ウィンドウ処理はこの値で、親ウィンドウに送るメッセージを確認する。

戻り値

正常終了	ステータスウィンドウのハンドル
異常終了	NULL

解　説

CreateStatusWindow関数は、アプリケーションのステータスを表示するステータスウィンドウを作成する。このウィンドウは、通常親ウィンドウのボトムに表示される。CreateStatusWindow関数は、ウィンドウを作成するためCreateWindow関数を呼ぶ。CreateWindowに変更なしでパラメータを渡す。そして配置、幅、高さのパラメータにデフォルト値をセットする。Windows 95では、最大16,364個のウィンドウハンドルがサポートされる。

参　照

→CreateWindow

DrawStatusText	――――	95	NT

ステータスウィンドウテキスト描画

V　C

```
void DrawStatusText(
    HDC       hdc,      //ウィンドウの表示コンテキストのハンドル
    LPRECT    lprc,     //RECT構造体のポインタ
    LPCTSTR   pszText,  //表示するテキストのポインタ
    UINT      uFlags    //描画フラグ
    );
```

パラメータ

- *hdc*　ウィンドウの表示コンテキストのハンドル。
- *lprc*　テキストが描かれる矩形のカレント座標での配置を含む、RECT構造体のポインタ。この関数は、矩形を指定する縁の内側に境界線を描く。
- *pszText*　表示するテキストを指定するNULLで終わる文字列へのポインタ。文字列のタブ文字は左詰め、右詰め、中央揃えの文字列を終わらせる。
- *uFlags*　テキストの描画フラグ。このパラメータは以下の値の組み合わせである。

値	意味
SBT_NOBORDERS	指定されたテキストのまわりに境界線を描かない。
SBT_POPOUT	テキストを目立たせるため高輝度の境界線を描く。
SBT_RTLREADING	Windows 95でのみ指定される。ヘブライ語やアラビア語のシステムで右から左へ読むテキストを表示する。

解　説

DrawStatusText関数は、ステータスウィンドウの境界線のスタイルに指定されたテキストを描く。

参　照

→RECT

MenuHelp	――――――	95	NT
ステータスウィンドウヘルプ表示			

V　C

```
void MenuHelp(
    UINT        uMsg,        //メッセージ
    WPARAM      wParam,      //1番目のメッセージパラメータ
    LPARAM      lParam,      //2番目のメッセージパラメータ
    HMENU       hMainMenu,   //メインメニューのハンドル
    HINSTANCE   hInst,       //モジュールのハンドル
    HWND        hwndStatus,  //ステータスウィンドウのハンドル
    UINTFAR*    lpwIDs       //文字列リソースの識別子とメニューハンドルの配
                               列のポインタ
    );
```

パラメータ

- *uMsg*　WM_MENUSELECTまたはWM_COMMANDメッセージ。
- *wParam*　1番目のメッセージパラメータ。
- *lParam*　2番目のメッセージパラメータ。
- *hMainMenu*　アプリケーションのメインメニューのハンドル。
- *hInst*　文字列リソースを含むモジュールのハンドル。
- *hwndStatus*　ステータスウィンドウのハンドル。
- *lpwIDs*　文字列リソースの識別子とメニューハンドルの組み合わせの配列へのポインタ。この関数は、指定されたメニューのハンドルを配列から検索し、妥当なヘルプ文字列をロードするため、対応するリソース識別子を使う。

解　説

MenuHelp関数は、WM_MENUSELECTとWM_COMMANDメッセージを処理し、指定されたステータスウィンドウのカレントメニューのヘルプテキストを表示する。

参　照

→WM_COMMAND,→WM_MENUSELECT

Toolbar

CreateMappedBitmap	————	95	NT
ツールバービットマップ作成			

V C

```
HBITMAP CreateMappedBitmap(
    HINSTANCE   hInstance,    //モジュール要求のハンドル
    int         idBitmap,     //ビットマップリソース識別子
    UINT        wFlags,       //ビットマップフラグ
    LPCOLORMAP  lpColorMap,   //COLORMAP構造体のポインタ
    int         iNumMaps      //カラーマップ番号
    );
```

パラメータ

・*hInstance* ビットマップリソースを含む実行ファイルのモジュール要求のハンドル。

・*idBitmap* ビットマップリソースのリソース識別子。

・*wFlags* ビットマップフラグ。このパラメータは0または、以下の値である。

値	意味
CMB_MASKED	マスクとしてビットマップを使う。

・*lpColorMap* ビットマップに必要な色の情報を含むCOLORMAP構造体へのポインタ。このパラメータがNULLの場合は、デフォルトカラーマップを使う。

・*iNumMaps* *lpColorMap*により指定されたカラーマップ番号。

戻り値

正常終了	ビットマップのハンドル
異常終了	NULL

解　説

CreateMappedBitmap関数は、ツールバーで使うビットマップを作成する。

CreateToolbarEx	――――――――	95	NT
ツールバーボタン作成			

V　C

```
HWND CreateToolbarEx(
    HWND          hwnd,         //親ウィンドウのハンドル
    DWORD         ws,           //ツールバーのウィンドウスタイル
    UINT          wID,          //ツールバーのコントロール識別子
    int           nBitmaps,     //ボタンイメージ番号
    HINSTANCE     hBMInst,      //モジュール要求
    UINT          wBMID,        //ビットマップリソース識別子
    LPCTBBUTTON   lpButtons,    //TBBUTTON構造体の配列へのポインタ
    int           iNumButtons,  //ボタン番号
    int           dxButton,     //ボタンの幅
    int           dyButton,     //ボタンの高さ
    int           dxBitmap,     //ボタンイメージの幅
    int           dyBitmap,     //ボタンイメージの高さ
    UINT          uStructSize   //TBBUTTON構造体の大きさ
    );
```

パラメータ

・*hwnd*　ツールバーの親ウィンドウのハンドル。

・*ws*　ツールバーのウィンドウスタイル。このパラメータは少なくともWS_CHILDスタイルを指定しなければならない。ツールバーには、スタイルの組み合わせを含むことができる。また、トピックスを関連付けることができる。

値	意味
WS_BORDER	境界を持つウィンドウを作成。
WS_CAPTION	タイトルバーを持つウィンドウを作成（暗黙に WS_BORDER スタイルを持つ)。WS_DLGFRAME スタイルと一緒に使うことはできない。
WS_CHILD	子ウィンドウを作成。WS_POPUPスタイルと一緒に使うことはできない。
WS_CLIPCHILDREN	親ウィンドウの内部で描画するときに、子ウィンドウが占める領域を除外する。親ウィンドウを作成するときに使う。
WS_CLIPSIBLINGS	互いに関連する兄弟ウィンドウをクリップする。ある特定の兄弟ウィンドウが描画メッセージを受け取ったときに、WS_CLIPSIBLINGSスタイルは、更新される兄弟ウィンドウの領域からオーバラップするほかのすべての兄弟ウィンドウをクリップする（WS_CLIPSIBLINGSを指定しないで兄弟ウィンドウがオーバラップしていると、隣の兄弟ウィンドウのクライアント領域内を描画できるようになる)。必ずWS_CHILDスタイルと一緒に使う。

WS_DISABLED	初期状態で使用禁止のウィンドウを作成。
WS_DLGFRAME	二重境界を持ち、タイトルを持たないウィンドウを作成。
WS_GROUP	ユーザーが方向キーを使って次のコントロールに移動できるコントロールのグループの、最初のコントロールを指定。最初のコントロールの後にWS_GROUPを使って定義されるコントロールは、すべて同じグループに属す。次のWS_GROUPスタイルを持つコントロールはそれ以前のグループを終了し、次のグループを開始する。
WS_HSCROLL	水平スクロールバーを持つウィンドウを作成。
WS_MAXIMIZE	最大表示されたウィンドウを作成。
WS_MAXIMIZEBOX	最大表示ボタンを持つウィンドウを作成。
WS_MINIMIZE	初期状態でアイコン化されたウィンドウを作成。必ずWS_OVERLAPPEDスタイルと一緒に使う。
WS_MINIMIZEBOX	アイコン化ボタンを持つウィンドウを作成。
WS_OVERLAPPED	オーバラップウィンドウを作成。オーバラップウィンドウは、通常キャプションと境界を持つ。
WS_OVERLAPPEDWINDOW	WS_OVERLAPPED、WS_CAPTION、WS_SYSMENU、WS_THICKFRAME、WS_MINIMIZEBOX、WS_MAXIMIZEBOXスタイルを持つオーバラップウィンドウを作成。
WS_POPUP	ポップアップウィンドウを作成。WS_CHILDスタイルと一緒に使うことはできない。
WS_POPUPWINDOW	WS_BORDER、WS_POPUP、WS_SYSMENUスタイルを持つポップアップウィンドウを作成。コントロールメニューを可視にするには、WS_CAPTIONスタイルとWS_POPUPWINDOWを組み合わせなければならない。
WS_SYSMENU	タイトルバーにコントロールメニューボックスを持つウィンドウを作成。タイトルバーを持つウィンドウにだけ使う。
WS_TABSTOP	ユーザーがTABキーを使って移動できるコントロールの1つであることを指定。ユーザーがTABキーを押すと次にWS_TABSTOPスタイルを持つコントロールに移動できる。
WS_THICKFRAME	ウィンドウのサイズ変更に使うことができる、太い枠を持つウィンドウを作成。
WS_VISIBLE	初期状態で可視のウィンドウを作成。
WS_VSCROLL	垂直スクロールバーを持つウィンドウを作成。

- *wID*　ツールバーのコントロール識別子。
- *nBitmaps*　*hBMInst*や*wBMID*で指定されるビットマップのボタンイメージ番号。
- *hBMInst*　ビットマップリソースを含む実行ファイルのモジュール要求。
- *wBMID*　ビットマップリソースのリソース識別子。*hBMInst*がNULLの場合、このパラメータは妥当なビットマップハンドルでなければならない。

- *lpButtons*　ツールバーに加えるボタンの情報を含むTBBUTTON構造体の配列へのポインタ。
- *iNumButtons*　ツールバーに加えるボタン番号。
- *dxButton*　ツールバーに加えるボタンの幅(ピクセル)。
- *dyButton*　ツールバーに加えるボタンの高さ(ピクセル)。
- *dxBitmap*　ツールバーに加えるボタンイメージの幅(ピクセル)。
- *dyBitmap*　ツールバーに加えるボタンイメージの高さ(ピクセル)。
- *uStructSize*　TBBUTTON構造体の大きさ。

戻り値

正常終了	ツールバーウィンドウのハンドル
異常終了	NULL

解説

CreateToolbarEx関数は、ツールバーウィンドウを作成し、ツールバーに指定されたボタンを加える。Windows 95では、最大16,364個のウィンドウハンドルをサポートする。

参照

→TBBUTTON

Up-Down Control

CreateUpDownControl

CreateUpDownControl	────	95	NT
アップダウンコントロール作成			

VC

```
HWND CreateUpDownControl(
    DWORD       dwStyle,    //コントロールのウィンドウスタイル
    int         x,          //水平座標
    int         y,          //垂直座標
    int         cx,         //上下コントロールの幅
    int         cy,         //上下コントロールの高さ
    HWND        hParent,    //親ウィンドウのハンドル
    int         nID,        //上下コントロールの識別子
    HINSTANCE   hInst,      //モジュール要求ハンドル
    HWND        hBuddy,     //上下コントロールに関連するウィンドウのハンドル
    int         nUpper,     //上下コントロールの上限
    int         nLower,     //上下コントロールの下限
    int         nPos        //コントロールの配置
);
```

パラメータ

- *dwStyle*　コントロールのウィンドウスタイル。このパラメータはWS_CHILD、WS_BORDER、WS_VISIBLEスタイルを含むが、上下コントロールを指定するウィンドウスタイルは、含まない。指定できる値はAjustWindowRectを参照。

- *x*　クライアント座標の中のコントロールの左上隅の水平座標。
- *y*　クライアント座標の中のコントロールの左上隅の垂直座標。
- *cx*　上下コントロールの幅(ピクセル)。
- *cy*　上下コントロールの高さ(ピクセル)。
- *hParent*　上下コントロールの親ウィンドウのハンドル。
- *nID*　上下コントロールの識別子。
- *hInst*　上下コントロールを作成するアプリケーションのモジュール要求ハンドル。
- *hBuddy*　上下コントロールに関連するウィンドウのハンドル。このパラメータがNULLのとき、コントロールには関連したウィンドウがない。
- *nUpper*　上下コントロールの上限。
- *nLower*　上下コントロールの下限。
- *nPos*　コントロールの配置。

戻り値

正常終了	アップダウンコントロールのハンドル
異常終了	NULL

解　説

CreateUpDownControl関数は、上下コントロールを作成する。Windows 95では、最大16,364個のウィンドウハンドルをサポートする。

Button

CheckDlgButton

CheckDlgButton	user32.dll	95	NT
ボタンコントロールチェック属性変更			

V　C

```
BOOL CheckDlgButton(
    HWND    hDlg,         //ダイアログボックスのハンドル
    int     nIDButton,    //ボタンコントロール識別子
    UINT    uCheck        //チェック状態
    );
```

V　B

```
Declare Function CheckDlgButton Lib "user32" Alias "CheckDLGButtonA" (ByVal hDlg As Long, ByVal nIDButton As Long, ByVal wCheck As Long) As Long
```

パラメータ

- *hDlg*　ボタンを含むダイアログボックスの識別子。
- *nIDButton*　変更するボタンの識別子。
- *uCheck*　ボタンのチェック状態を指定する。このパラメータは次の値の１つを指定できる。

値	意味
0	ボタンがチェックされていない。Windows 95のBST_UNCHECKEDと同様。
1	ボタンがチェックされている。Windows 95のBST_CHECKEDと同様。
2	ボタンが淡色表示されている(BS_3STATEまたは、BS_AUTO3STATEスタイルの場合のみ)。Windows 95のBST_INDETERMINATEと同様。
BST_CHECKED	Windows 95のみ。ボタンがチェックされている。
BST_INDETERMINATE	Windows 95のみ。ボタンが淡色表示されている。(BS_3STATEまたは、BS_AUTO3STATEスタイルの場合)。
BST_UNCHECKED	Windows 95のみ。ボタンがチェックされていない。

戻り値

正常終了	TRUE
異常終了	FALSE

詳細なエラー情報はGetLastError関数で取得する。

解　説

CheckDlgButton関数は、ボタンコントロールを選択(ボタンコントロールの隣りにチェックマークを付加)または選択解除(チェックマークを除去)する。3ステートボタン(3種類の状態を持つ)の状態を変更するときにもこの関数を使う。CheckDlgButton関数は、指定されたダイアログボックス内にある指定されたボタンコントロールに、BM_SETCHECKメッセージを送る。

参　照

→CheckRadioButton,→IsDlgButtonChecked

CheckRadioButton

user32.dll　95　NT

ラジオボタンチェック属性変更

V　C

```
BOOL CheckRadioButton(
    HWND  hDlg,            //ダイアログボックスのハンドル
    int   nIDFirstButton,  //グループ内の最初のラジオボタン識別子
    int   nIDLastButton,   //グループ内の最後のラジオボタン識別子
    int   nIDCheckButton   //選択するラジオボタン識別子
);
```

V　B

Declare Function CheckRadioButton Lib "user32" Alias "CheckRadioButtonA" (ByVal hDlg As Long, ByVal nIDFirstButton As Long, ByVal nIDLastButton As Long, ByVal nIDCheckButton As Long) As Long

パラメータ

- *hDlg*　ラジオボタンを含むダイアログボックスの識別子。
- *nIDFirstButton*　グループ内にある最初のラジオボタンの識別子を指定する。
- *nIDLastButton*　グループ内にある最後のラジオボタンの識別子を指定する。
- *nIDCheckButton*　選択するラジオボタンの識別子を指定する。

戻り値

正常終了	TRUE
異常終了	FALSE

詳細なエラー情報はGetLastError関数により取得する。

解　説

CheckRadioButton関数は、グループ内の指定されたラジオボタンにチェックマークを付加し、グループ内のそのほかのすべてのラジオボタンのチェックマークを取り消す。CheckRadioButton関数は、指定されたグループにあるラジオボタンに、BM_SETCHECKメッセージを送る。

参　照

→BM_SETCHECK,→CheckDlgButton,→IsDlgButtonChecked

IsDlgButtonChecked | user32.dll | 95 | NT

ボタンコントロール状態取得

V　C

```
UINT IsDlgButtonChecked(
        HWND  hDlg,         //ダイアログボックスハンドル
        int   nIDButton     //ボタン識別子
        );
```

V　B

Declare Function IsDlgButtonChecked Lib "user32" Alias "IsDlgButtonChecked" (ByVal hDlg As Long, ByVal nIDButton As Long) As Long ByVal nIDLastButton As Long, ByVal nIDCheckButton As Long) As Long

パラメータ

- *hDlg*　ボタンコントロールを含むダイアログボックスの識別子。
- *nIDButton*　ボタンコントロールの識別子を指定する。

戻り値

BS_AUTOCHECKBOX、BS_AUTORADIOBUTTON、BS_AUTO3STATE、BS_CHECKBOX、BS_RADIOBUTTON、BS_3STATEスタイルのボタンの場合の戻り値は、以下の値の1つである。

値	意味
0	ボタンがチェックされていない。Windows 95のBST_UNCHECKEDと同様。
1	ボタンがチェックされている。Windows 95のBST_CHECKEDと同様。
2	ボタンが淡色表示されている(BS_3STATEまたは、BS_AUTO3STATEスタイルの場合)。Windows 95のBST_INDETERMINATEと同様。
BST_CHECKED	Windows 95のみ。ボタンがチェックされている。
BST_INDETERMINATE	Windows 95のみ。ボタンが淡色表示されている(BS_3STATEまたは、BS_AUTO-3STATEスタイルの場合)。
BST_UNCHECKED	Windows 95のみ。ボタンがチェックされていない。

解説

IsDlgButtonChecked関数は、ボタンがチェックされているかどうか、また3ステートボタンが淡色表示、チェック、またはそのどちらでもない状態のいずれであるかを判断する。IsDlgButtonChecked関数は、指定されたボタンコントロールにBM_GETCHECKメッセージを送る。

参照

→CheckDlgButton

Combo Box

DlgDirListComboBox

DlgDirListComboBox	user32.dll	95	NT

パス名と一致するファイル名をコンボボックス内に一覧表示

VC

```
int DlgDirListComboBox(
    HWND    hDlg,          //コンボボックスを持つダイアログボックスのハンドル
    LPTSTR  lpPathSpec,    //パス名またはファイル名の文字列ポインタ
    int     nIDComboBox,   //コンボボックスの識別子
    int     nIDStaticPath, //スタティックコントロールの識別子
    UINT    uFiletype      //表示するファイル属性
    );
```

VB

```
Declare Function DlgDirListComboBox Lib "user32" Alias "DlgDirListComboBoxA" (ByVal hDlg As Long, ByVal lpPathSpec As String, ByVal nIDComboBox As Long, ByVal nIDStaticPath As Long, ByVal wFileType As Long) As Long
```

パラメータ

- *hDlg*　コンボボックスを含むダイアログボックスの識別子。
- *lpPathSpec*　NULLで終わる以下の形式の文字列ポインタ。[drive:][[\]directory[\directory]...\][filename]文字列にドライブ名やディレクトリ名を指定したときは、DlgDirListComboBox関数は、リストボックスに格納する前にカレントドライブとカレントディレクトリを変更する。リストが格納されると、ドライブ名やディレクトリ名の部分が削除されて、*lpPathSpec*パラメータが更新される。
- *nIDComboBox*　ダイアログボックスのコンボボックスコントロールの識別子を指定する。このパラメータが0のとき、DlgDirListComboBox関数はリストボックスが存在しないものとし、ファイル名やディレクトリを格納しない。
- *nIDStaticPath*　カレントディレクトリの表示のために使われるスタティックテキストコントロールの識別子を指定する。このパラメータが0のとき、DlgDirListComboBox関数はそのコントロールが存在しないものとする。

・*uFiletype* 表示するファイル名のファイル属性を指定する。このパラメータは、以下の値の組み合わせになる。

値	意味
DDL_ARCHIVE	アーカイブファイル。
DDL_DIRECTORY	サブディレクトリ。サブディレクトリ名は角括弧 ([ ]) の中。
DDL_DRIVES	ドライブ。ドライブは[-x-]の形式で格納される。xはドライブの文字。
DDL_EXCLUSIVE	指定された属性を持つファイル。デフォルトは、読み書きファイル。
DDL_HIDDEN	隠しファイル。
DDL_READONLY	読み取り専用ファイル。
DDL_READWRITE	ほかの属性を持たない、読み書き可能なファイル。
DDL_SYSTEM	システムファイル。
DDL_POSTMSGS	ポストメッセージをアプリケーションキューに送る。デフォルトは、DlgDirList関数がダイアログボックスプロシージャに直接メッセージを送る。

戻り値

正常終了	0以外
異常終了	0

たとえば、*lpPathSpec*で指定される文字列が不正なパスであった場合、この関数は失敗する。詳細なエラー情報はGetLastError関数により取得する。

解　説

関数は、指定されたコンボボックスにディレクトリのリストを格納する。この関数は、指定された属性やパス名と一致するすべてのファイルやドライブ名をリストに格納する。*lpPathSpec*パラメータポインタが長さが0の文字列を指していたり、ドライブ名やディレクトリ名が指定されたが、ファイル名が無い場合は、ファイル名は*.*(ワイルドカード)が指定されたとする。WindowsNTでは、ディレクトリリストは長いファイル名を表示する。Windows 95では、ディレクトリリストは短いファイル名を表示する(8.3フォーム)。SHGetFileInfoやGetFullPathName関数を使用し対応する長いファイル名を取得する。

参　照

→DlgDirList,→DlgDirSelectComboBoxEx,→GetFullPathName,GetFileInfo

DlgDirSelect	―――――――	95	NT

リストボックスから現在の選択項目を文字列バッファにコピーする

V　C

```
BOOL DlgDirSelect(
    HWND  hwndDlg,    //リストボックスを持つダイアログボックスのハンドル
    LPSTR lpszPath,   //パス名またはファイル名の文字列バッファのアドレス
    int   idListBox   //リストボックスの識別子
    );
```

パラメータ

- *hwndDlg*　リストボックスを持つダイアログボックスの識別子。
- *lpszPath*　パス名またはファイル名を格納する、128バイトのバッファへのポインタ。
- *idListBox*　ダイアログボックス内のリストボックスの識別子を表す整数を指定する。

戻り値

正常終了	0以外の値
異常終了	0

解　説

DlgDirSelect関数は、リストボックスで現在選択されている項目を取得する。この関数は、DlgDirList関数によりリストボックスにデータが格納されていることを前提として動作する。選択項目は、ドライブ名、ファイル名、またはディレクトリ名として処理される。

現在選択されている項目がディレクトリ名またはドライブ名のとき、DlgDirSelect関数は、それらの名前から角括弧やハイフンを取り除いて、新しいパス名に挿入できるようにする。何も選択されていないときには、*lpszPath*パラメータが指すバッファの内容は変わらない。

DlgDirSelect関数を使用して、リストボックスから複数のファイル名を得ることはできない。複数の項目を選択できるリストボックスではこの関数を使えない。誤って使用した場合、この関数は0を返さず、*lpszPath*パラメータの指す文字列は変更されない。

DlgDirSelect関数は、LB_GETCURSELメッセージとLB_GETTEXTメッセージをリストボックスに送る。

DlgDirSelect関数は、Win32 API処理ではない。Win32ベースのアプリケーションはDlgDirSelectEx関数を使用する。

参　照

DlgDirList,→DlgDirListComboBox,→DlgDirSelectComboBox,DlgDirSelectEx

DlgDirSelectComboBox	————————	95	NT

DlgDirSelectComboBox

コンボボックスから現在の選択項目を文字列バッファにコピーする

V C

```
BOOL DlgDirSelectComboBox(
    HWND   hwndDlg,       //リストボックスを持つダイアログボックスのハンドル
    LPSTR  lpszPath,      //パス名またはファイル名の文字列バッファのポインタ
    int    idComboBox     //コンボボックスの識別子
    );
```

パラメータ

- *hwndDlg*　コンボボックスを持つダイアログボックスの識別子。
- *lpszPath*　パス名またはファイル名を格納する、128バイトのバッファを指すポインタ。
- *idComboBox*　ダイアログボックス内のコンボボックスの識別子を表す整数を指定する。

戻り値

正常終了	0以外の値
異常終了	0

解　説

DlgDirSelectComboBox関数は、コンボボックスのリストボックスで現在選択されている項目を取得する。この関数は、DlgDirListComboBox関数によりリストボックスにデータが格納されていることを前提として動作する。選択項目は、ドライブ名、ファイル名、またはディレクトリ名として処理される。

DlgDirSelectComboBox関数を使用して、コンボボックスから複数のファイル名を得ることはできない。

現在選択されている項目がディレクトリ名またはドライブ名のとき、DlgDirSelectComboBox関数は、それらの名前から角括弧やハイフンを取り除いて、新しいパス名に挿入できるようにする。何も選択されていないときには、*lpszPath*パラメータが指すバッファの内容は変わらない。

DlgDirSelectComboBox関数は、CB_GETCURSELメッセージとCB_GETLBTEXTメッセージをリストボックスに送る。

DlgDirSelectComboBox関数は、Win32 API処理ではない。Win32ベースのアプリケーションはDlgDirSelectComboBoxEx関数を使用する。

Dialog Box

CreateDialog

CreateDialog	――――	95	NT

モードレスダイアログボックス作成

V C

```
HWND CreateDialog(
    HINSTANCE hInstance,     //アプリケーションインスタンスのハンドル
    LPCTSTR   lpTemplate,    //ダイアログボックスのテンプレート名の識別子
    HWND      hWndParent,    //オーナーウィンドウのハンドル
    DLGPROC   lpDialogFunc   //ダイアログボックスプロシージャのインスタン
                               スポインタ
);
```

パラメータ

- *hInstance*　ダイアログボックステンプレートを含む実行可能ファイルのあるモジュールのインスタンスの識別子。
- *lpTemplate*　ダイアログボックステンプレートの識別子。ダイアログボックステンプレートの名前を指す、NULLで終わる文字列のポインタ。このパラメータがリソース識別子を指定した場合、上位ワードは0でなければならない。下位ワードは識別子を含まなければならない。MAKEINTRESOURCEマクロでこの値を作成する。
- *hWndParent*　ダイアログボックスを所有するウィンドウの識別子。
- *lpDialogFunc*　ダイアログボックスプロシージャのポインタ。ダイアログボックスプロシージャについて詳しくは、DialogProc関数を参照すること。

戻り値

正常終了	ダイアログボックスのハンドル
異常終了	NULL

解　説

CreateDialog関数は、ダイアログボックステンプレートリソースから、モードレスダイアログボックスを作成する。ダイアログボックスを作成するときには、CreateWindowEx関数が呼び出される。ダイアログボックスプロシージャはWM_INITDIALOGメッセージ(DS_SETFONTスタイルが指定された場合は、WM_SETFONTメッセージ)を受け取り、ダイアログボックスが表示される。ダイアログボックステンプレートでWS_VISIBLEスタイルを指定すれば、ダイアログボックスが表示される。CreateDialog関数は、ダイアログボックスを作成するとすぐに制御を戻す。
CreateDialog関数を使用して作成したダイアログボックスを破棄するには、DestroyWindow関数を使う。Windows 95では、最大16,364個までのウィンドウハンドルをサポートしている。

参　照

CreateDialogIndirect,CreateDialogIndirectParam,CreateDialogParam,CreateWindowEx,
→DestroyWindow,→DialogBox,→DialogProc,ShowWindow,→WM_INITDIALOG,
→WM_SETFONT

DefDlgProc	user32.dll	95	NT
ダイアログボックスメッセージ処理			

V C

```
LRESULT DefDlgProc(
    HWND      hDlg,     //ダイアログボックスのハンドル
    UINT      uMsg,     //メッセージ
    WPARAM    wParam,   //第1メッセージパラメータ
    LPARAM    lParam    //第2メッセージパラメータ
    );
```

V B

Declare Function DefDlgProc Lib "user32" Alias "DefDlgProcA" (ByVal hDlg As Long, ByVal wMsg As Long, ByVal wParam As Long, ByVal lParam As Long) As Long

パラメータ

- *hDlg* ダイアログボックスの識別子。
- *uMsg* メッセージ番号を指定する。
- *wParam* メッセージにより異なる追加情報を指定する。
- *lParam* メッセージにより異なる追加情報を指定する。

戻り値

メッセージ処理の結果を返す。結果の意味は送られたメッセージにより異なる。

解説

DefDlgProc関数は、アプリケーション定義のダイアログボックスクラスに属するウィンドウ処理のためのデフォルトメッセージ処理を行う。
DefDlgProc関数はダイアログボックスのあらかじめ定義されたクラスのためのウィンドウプロシージャである。アプリケーションの中でダイアログボックスの機能を継承する新しいウィンドウクラスを作成するには、この関数を使用しなければならない。
DefDlgProc関数は、ダイアログボックスプロシージャ内のメッセージを処理するデフォルトのハンドラとして呼び出されるようなことはない。そのようにして呼び出した場合、実行が再帰的になってしまう。

参照

DefWindowProc,GetClassInfo,RegisterClass,→WNDCLASS

DialogBox	――――――――	95	NT
モーダルダイアログボックス作成			

V　C

```
int DialogBox(
        HANDLE      hInstance,     //アプリケーションインスタンスのハンドル
        LPCTSTR     lpTemplate,    //ダイアログボックステンプレート名のポインタ
        HWND        hWndParent,    //オーナーウィンドウのハンドル
        DLGPROC     lpDialogFunc   //ダイアログボックスプロシージャのポインタ
        );
```

パラメータ

- *hInstance*　ダイアログボックステンプレートを含む実行可能ファイルのモジュールのインスタンスの識別子。
- *lpTemplate*　ダイアログボックステンプレートの識別子。ダイアログボックステンプレートの名前を指すNULLで終わる文字列を指すポインタか、ダイアログボックステンプレートのリソース識別子を示す整数値である。上位ワードは、0でなければならない。下位ワードは識別子を含まなければならない。MAKEINTRESOURCEマクロを使いこの値を作成する。
- *hWndParent*　ダイアログボックスを所有するウィンドウの識別子。
- *lpDialogFunc*　ダイアログボックスプロシージャのポインタ。ダイアログボックスプロシージャについて詳しい情報は、DialogProcコールバック関数を参照のこと。

戻り値

正常終了	ダイアログボックスを終了時に利用するEndDialog関数で指定した*nResult*パラメータの値
異常終了	-1

解　説

DialogBox関数は、ダイアログボックステンプレートリソースからモーダルダイアログボックスを作成する。DialogBox関数は、ダイアログボックスプロシージャがEndDialog関数を呼び出してモーダルダイアログボックスを終了するまで、制御を戻さない。ダイアログボックスの作成時にはCreateWindowEx関数が呼び出される。その後ダイアログボックスプロシージャはWM_INITDIALOGメッセージ(DS_SETFONTスタイルが指定されているときはWM_SETFONTメッセージ)を受け取り、ダイアログボックスが表示される。
Windows 95では、システムが最大16,364個のウィンドウハンドルをサポートする。

参　照

→CreateDialog,CreateWindowEx,DialogBoxIndirect,DialogBoxIndirectParam,DialogBoxParam,→DialogProc,→EndDialog,→MAKEINTRESOURCE,
→WM_INITDIALOG,→WM_SETFONT

DialogProc	————	95	NT
ダイアログボックスメッセージ処理			

V C

```
BOOL CALLBACK DialogProc(
    HWND      hwndDlg,   //ダイアログボックスのハンドル
    UINT      uMsg,      //メッセージ
    WPARAM    wParam,    //第1メッセージパラメータ
    LPARAM    lParam     //第2メッセージパラメータ
    );
```

パラメータ

- *hwndDlg*　ダイアログボックスの識別子。
- *uMsg*　メッセージを指定する。
- *wParam*　メッセージにより異なる追加情報を指定する。
- *lParam*　メッセージにより異なる追加情報を指定する。

戻り値

WM_INITDIALOGメッセージへの応答の場合を除いて、ダイアログボックスプロシージャは、メッセージを処理したときは0以外の値を、それ以外のときは0を返さなければならない。WM_INITDIALOGメッセージに応答した場合、ダイアログボックスプロシージャはSetFocus関数を呼び出してダイアログボックスのコントロールの1つにフォーカスを設定したときは、0を返されなければならない。それ以外のときは、0以外の値を返さなければならない。この場合、ダイアログボックス内のフォーカスを与えることのできる最初のコントロールに、システムがフォーカスを設定する。

解　説

DialogProc関数は、モーダルやモードレスダイアログボックスに送られたメッセージを処理する、アプリケーション定義のコールバック関数である。
ダイアログボックスプロシージャは、そのダイアログボックスのためにダイアログボックスクラスが利用されるときだけ使われる。このクラスはデフォルトのクラスであり、ダイアログボックステンプレートで明確にクラスが指定されなかったときに使われる。ダイアログボックスプロシージャはウィンドウプロシージャに似ているが、必要としないメッセージを処理するときにDefWindowProc関数を呼び出してはならない。不要なメッセージはダイアログボックスウィンドウプロシージャが内部的に処理する。
DialogProcは、アプリケーション定義の関数名のプレースホルダである。

参　照

→CreateDialog,CreateDialogIndirect,CreateDialogIndirectParam,CreateDialogParam,
DefWindowProc,DialogBox,DialogBoxIndirect,DialogBoxIndirectParam,
DialogBoxParam,SetFocus,→WM_INITDIALOG

EndDialog	user32.dll	95	NT
モーダルダイアログボックス消去			

V　C

```
EndDialog(
    HWND    hDlg,       //ダイアログボックスのハンドル
    int     nResult     //戻り値
    );
```

V　B

Declare Function EndDialog Lib "user32" Alias "EndDialog" (ByVal hDlg As Long, ByVal nResult As Long) As Long

パラメータ

・*hDlg*　破棄されるダイアログボックスの識別子。
・*nResult*　ダイアログボックスを作成した関数からアプリケーションに返す値を指定する。

戻り値

正常終了	TRUE
異常終了	FALSE

解　説

EndDialog関数は、モーダルダイアログボックスを消去して、ダイアログボックス処理を終了しシステムに復帰させる。
EndDialog関数は、DialogBox,DialogBoxParam,DialogBoxIndirect,そしてDialogBoxIndirectParam関数により作成されたダイアログボックスの処理を完了するときに使う。アプリケーションはEndDialog関数を、ダイアログボックスプロシージャ内から呼び出す。ダイアログボックスプロシージャは、WM_INITDIALOGメッセージの処理中でもEndDialog関数を呼び出すことができる。WM_INITDIALOGメッセージの処理の途中で呼び出された場合、ダイアログボックスは表示される前、またはフォーカスが設定される前に、消去される。
EndDialog関数は、ダイアログボックスをすぐに破棄するのではなく、DialogBox関数の実行が終了した直後にダイアログボックスを破棄するようシステムに指示するフラグを設定する。

参　照

→DialogBox,DialogBoxIndirect,DialogBoxIndirectParam,DialogBoxParam,
→WM_INITDIALOG

GetDialogBaseUnits	user32.dll	95	NT
ダイアログボックスベース単位取得			

V C LONG GetDialogBaseUnits();

V B Declare Function GetDialogBaseUnits Lib "user32" Alias "GetDialogBaseUnits" () As Long

戻り値

上位ワード	ダイアログボックスの高さ
下位ワード	ダイアログボックスのベース幅

解 説

GetDialogBaseUnits関数は、Windowsがダイアログボックスを作成するときに使うダイアログボックスのベース単位を返す。Windowsとアプリケーションはこの値を使用して、システムフォントの文字の平均幅を計算する。返される値は、ダイアログボックス単位にスケーリングされる前のダイアログボックスのベース単位を示す。x方向のダイアログボックス単位は、この関数が返す幅の4分の1である。y方向のダイアログボックス単位は、この関数が返す高さの8分の1である。

GetDialogBaseUnits関数を使用してコントロールの高さと幅(ピクセル単位)を決めるときには、ダイアログボックス単位の幅(x)と高さ(y)、および戻り値(lDlgBaseUnits)を以下の公式に代入する。

```
(x*LOWORD(lDlgBaseUnits)) / 4
(y*HIWORD(lDlgBaseUnits)) / 8
```

以下の公式はアプリケーションでピクセルからダイアログ単位を変換する。

```
dialogunitX=(pixelX*4) / baseunitX
dialogunitY=(pixelY*8) / baseunitY
```

ダイアログのベース単位が4で割り切れない値のときは、丸め誤差を避けるために、割り算の前に適切な定数を掛ける。

参 照 →MapDialogRect

GetDlgCtrlID	user32.dll	95	NT
コントロール識別子取得			

V　C

```
int GetDlgCtrlID(
        HWND        hwndCtl    //コントロールのハンドル
        );
```

V　B

Declare Function GetDlgCtrlID Lib "user32" Alias "GetDlgCtrlID" (ByVal hwnd As Long) As Long

パラメータ

・*hwndCtl*　　コントロールの識別子。

戻り値

正常終了	コントロールの識別子
異常終了	NULL

たとえば、*hwndCtl*パラメータに不正な値がセットされた場合、この関数は失敗する。

解　説

GetDlgCtrlID関数は、指定されたコントロールの識別子を返す。この関数は、ダイアログボックス内のコントロールのハンドルだけでなく、任意の子ウィンドウのハンドルも返す。トップレベルウィンドウは識別子を持たないため、*hwndCtl*パラメータにトップレベルウィンドウを識別するハンドルを指定した場合、GetDlgCtrlID関数の戻り値は無効になる。

参　照

→CreateWindow,CreateWindowEx,→GetDlgItem

GetDlgItem	user32.dll	95	NT
コントロールハンドル取得			

V　C

```
HWND GetDlgItem(
        HWND hDlg,        //ダイアログボックスのハンドル
        int     nIDDlgItem //コントロールの識別子
        );
```

V　B

Declare Function GetDlgItem Lib "user32" Alias "GetDlgItem" (ByVal hDlg As Long, ByVal nIDDlgItem As Long) As Long

パラメータ

・*hDlg*　　コントロールを持つダイアログボックスの識別子。
・*nIDDlgItem*　　取得するコントロールの識別子を指定する。

戻り値

正常終了	コントロールのウィンドウハンドル
異常終了	NULL

解　説

GetDlgItem関数は、指定されたダイアログボックス内にあるコントロールのハンドルを取得する。

GetDlgItem関数は、ダイアログボックスだけでなく、任意の親ウィンドウと子ウィンドウのペアに対しても使える。子ウィンドウが一意の識別子(CreateWindowまたはCreateWindowEx関数を使用して子ウィンドウを作成したときに*hMenu*パラメータに指定された識別子)を持っていれば、*hDlg*パラメータに親ウィンドウを識別するハンドルを指定すると、GetDlgItem関数は子ウィンドウのハンドルを返す。

参　照　→CreateWindow,CreateWindowEx,GetDlgItemInt,GetDlgItemText

GetNextDlgGroupItem　user32.dll　95　NT
コントロールハンドル取得

V　C

```
HWND GetNextDlgGroupItem(
        HWND        hDlg,       //ダイアログボックスのハンドル
        HWND        hCtl,       //コントロールのハンドル
        BOOL        bPrevious   //方向フラグ
        );
```

V　B

```
Declare Function GetNextDlgGroupItem Lib "user32" Alias "GetNextDlgGroupItem"
(ByVal hDlg As Long, ByVal hCtl As Long, ByVal bPrevious As Long) As Long
```

パラメータ

- *hDlg*　検索されるダイアログボックスの識別子。
- *hCtl*　検索の開始配置にあるコントロールの識別子。このパラメータがNULLのとき、この関数は検索の開始配置としてダイアログボックスの最後(または最初)のコントロールを使う。
- *bPrevious*　ダイアログボックス内のコントロールのグループを関数がどのようにして検索するかを指定する。このパラメータがTRUEのときは、関数はグループ内の前のコントロールを検索する。FALSEのときは、グループ内の次のコントロールを検索する。

戻り値

正常終了	コントロールのウィンドウハンドル

解　説

GetNextDlgGroupItem関数は、ダイアログボックス内のコントロールのグループの範囲内で、前(または次)にあるコントロールを検索する。コントロールのグループ範囲は、WS_GROUPスタイルのコントロールからWS_GROUPスタイルを含まない最後のコントロールまでである。

*hCtl*パラメータがグループ内の最後のコントロールを識別して、*bPrevious*パラメータがFALSEのときには、GetNextDlgGroupItem関数はグループ内の最初のコントロールのウィンドウハンドルを返す。*hCtl*がグループ内の最初のコントロールを識別して、*bPrevious*がTRUEのときには、GetNextDlgGroupItem関数はグループ内の最後のコントロールのウィンドウハンドルを返す。

参　照　→GetNextDlgTabItem

GetNextDlgTabItem	user32.dll	95	NT
コントロールハンドル取得			

V　C

```
HWND GetNextDlgTabItem(
        HWND        hDlg,       //ダイアログボックスのハンドル
        HWND        hCtl,       //既知のコントロールのハンドル
        BOOL        bPrevious   //方向フラグ
        );
```

V　B

Function GetNextDlgTabItem Lib "user32" Alias "GetNextDlgTabItem" (ByVal hDlg As Long, ByVal hCtl As Long, ByVal bPrevious As Long) As Long

パラメータ

- *hDlg*　検索されるダイアログボックスの識別子。
- *hCtl*　検索の開始配置にあるコントロールの識別子。このパラメータがNULLのとき、この関数は検索の開始配置としてダイアログボックスの最後(または最初)のコントロールを使う。
- *bPrevious*　ダイアログボックスを関数がどのようにして検索するかを指定する。このパラメータがTRUEのときは、関数はダイアログボックス内の前のコントロールを検索する。

戻り値

正常終了	コントロールのウィンドウハンドル

解　説

GetNextDlgTabItem関数は、指定されたコントロールに先行する(または後に続く)WS_TABSTOPスタイルを持つ最初のコントロールのハンドルを取得する。

参　照

→GetDlgItem,→GetNextDlgGroupItem

IsDialogMessage	user32.dll	95	NT
ダイアログボックスメッセージ処理			

V C

```
BOOL IsDialogMessage(
    HWND      hDlg,    //ダイアログボックスのハンドル
    LPMSG     lpMsg    //メッセージの構造体のアドレス
    );
```

V B

Declare Function IsDialogMessage Lib "user32" Alias "IsDialogMessageA" (ByVal hDlg As Long, lpMsg As MSG) As Long

パラメータ

・*hDlg*　ダイアログボックスを識別するハンドル。
・*lpMsg*　チェックされるメッセージを含むMSG構造体のポインタ。

戻り値

正常終了	TRUE
異常終了	FALSE

関数は、指定されたメッセージが指定されたモードレスダイアログボックスに対するものならば、メッセージを処理する。

解　説

IsDialogMessageはモードレスダイアログボックス用だが、コントロールを含むウィンドウに対して使用すれば、ダイアログボックスと同じキーボード選択を実現できる。IsDialogMessageは、メッセージを処理するとき、キーボードメッセージをチェックして、対応するダイアログボックスの選択コマンドに変換する。たとえば、タブキーを押すと、次のコントロールまたはコントロールのグループが選択される。また、↓キーを押すと、グループの次のコントロールが選択される。IsDialogMessageで処理されたメッセージは、TranslateMessage関数やDispatchMessage関数に渡してはならない。これは、必要なメッセージの変換やディスパッチをIsDialogMessageがすべて実行するためである。IsDialogMessageは、処理するキーを判断するため、ダイアログボックスプロシージャにWM_GETDLGCODEメッセージを送る。IsDialogMessageは、ウィンドウにDM_GETDEFIDメッセージやDM_SETDEFIDメッセージを送る場合がある。これらのメッセージはWINDOWS.HヘッダーファイルでWM_USERとWM_USER+1として定義されているため、同じ値のアプリケーション定義のメッセージと衝突する可能性がある。

参　照

→DispatchMessage,→DM_GETDEFID,→DM_SETDEFID,→MSG,TranslateMessage,→WM_GETDLGCODE,→WM_USER

MapDialogRect	user32.dll	95	NT
ダイアログボックス単位変換			

V C

```
BOOL MapDialogRect(
    HWND     hDlg,     //ダイアログボックスのハンドル
    LPRECT   lpRect    //矩形の構造体のアドレス
    );
```

V B

Declare Function MapDialogRect Lib "user32" Alias "MapDialogRect" (ByVal hDlg As Long, lpRect As RECT) As Long

パラメータ

- *hDlg*　ダイアログボックスの識別子。このダイアログボックスは、ダイアログボックス作成関数を使用して作成されたものでなければならない。
- *lpRect*　変換されるダイアログボックスの座標を含む、RECT構造体のポインタ。

戻り値

正常終了	TRUE
異常終了	FALSE

解　説

MapDialogRect関数は、指定されたダイアログボックス単位をスクリーン単位(ピクセル)に変換(マップ)する。

MapDialogRect関数は、矩形のダイアログボックス単位をスクリーン単位に変換する。ダイアログボックス単位は、現在のダイアログの基本単位に相対的に定義される。ダイアログの基本単位は、ダイアログボックステキストに使われるフォントの平均文字幅と高さに基づいたものである。以下の公式がこの処理である。

```
left=(left*baseunitX) / 4
right=(right*baseunitX) / 4
top=(top*baseunitY) / 8
bottom=(bottom*baseunitY) / 8
```

一般に、ダイアログボックスはSystemフォントを使うが、リソース定義ファイルでDS_SETFONTスタイルを指定することにより、アプリケーションは別のフォントを指定することができる。

参　照

→CreateDialog,CreateDialogIndirect,CreateDialogIndirectParam,CreateDialogParam,
→DialogBox,DialogBoxIndirect,DialogBoxIndirectParam,DialogBoxParam,
GetDialogBaseUnits,→RECT

MessageBox	user32.dll	95	NT
メッセージボックス作成			

V C

```
int MessageBox(
        HWND        hWnd,       //オーナーウィンドウのハンドル
        LPCTSTR     lpText,     //メッセージボックスのテキストのアドレス
        LPCTSTR     lpCaption,  //メッセージボックスのタイトルのアドレス
        UINT        uType       //メッセージボックスのスタイル
        );
```

V B

```
Declare Function MessageBox Lib "user32" Alias "MessageBoxA" (ByVal hwnd As Long, ByVal lpText As String, ByVal lpCaption As String, ByVal wType As Long) As Long
```

パラメータ

- *hWnd* 作成するメッセージボックスの所有ウィンドウの識別子。このパラメータがNULLのときは、メッセージボックスに所有ウィンドウはない。
- *lpText* 表示されるメッセージを指定するNULLで終わる文字列のポインタ。
- *lpCaption* ダイアログボックスのタイトルを指定するNULLで終わる文字列のポインタ。このパラメータがNULLのときは、デフォルトタイトルの[エラー]が使われる。
- *uType* ダイアログボックスの内容と動作を指定する。このパラメータは、以下の値の組み合わせになる。

値	意味
MB_ABORTRETRYIGNORE	メッセージボックスには、[中止]、[再試行]、[無視]の3つのプッシュボタンが含まれる。
MB_APPLMODAL	ユーザーは、*hWnd*パラメータが識別するウィンドウ内で作業を継続する前に、メッセージボックスに応答しなければならない。しかし、ユーザーは、ほかのアプリケーションのウィンドウに移動し、そのウィンドウ内で作業することができる。MB_SYSTEM MODALMB_TASKMODALを指定しなければ、MB_APPLMODALがデフォルトである。
MB_DEFAULT_DESKTOP_ONLY	入力を受け取るデスクトップはデフォルトデスクトップでなければならない。その他の場合、この関数は失敗する。デフォルトデスクトップは、ユーザーがログオンした後に実行されるアプリケーションである。
MB_DEFBUTTON1	最初のボタンがデフォルトになる。MB_DEF BUTTON2 、MB_DEFBUTTON3を指定しなければ、最初のボタンが常にデフォルトになる。
MB_DEFBUTTON2	2番目のボタンがデフォルトになる
MB_DEFBUTTON3	3番目のボタンがデフォルトになる。
MB_DEFBUTTON4	4番目のボタンがデフォルトになる。

MB_HELP	Windows 95では、メッセージボックスにヘルプボタンが加わる。ヘルプボタンかF1を押すとヘルプが表示される。
MB_ICONASTERISK	MB_ICONINFORMATIONと同じ。
MB_ICONERROR	Windows 95では、MB_ICONHANDと同じに使用される。
MB_ICONEXCLAMATION	感嘆符アイコンがメッセージボックスに表示される。
MB_ICONHAND	MB_ICONSTOPと同じ。
MB_ICONINFORMATION	円の中に小文字の「i」があるアイコンがメッセージボックス内に表示される。MB_ICONQUESTION疑問符のアイコンがメッセージボックス内に表示される。
MB_ICONSTOP	「STOP」アイコンがメッセージボックスに表示される。
MB_ICON_WARNING	Windows 95では、MB_ICONEXCLAMATIONと同じに使用される。
MB_OK	メッセージボックスには、[OK]プッシュボタンが含まれる。
MB_OKCANCEL	メッセージボックスには、[OK]プッシュボタンと[キャンセル]プッシュボタンが含まれる。
MB_RETRYCANCEL	メッセージボックスには、[再試行]プッシュボタンと[キャンセル]プッシュボタンが含まれる。
MB_RIGHT	Windows 95では、テキストの右詰め。
MB_RTLREADING	Windows 95では、ヘブライ語やアラビア語で使用される右から左へ読むテキストのメッセージを表示。
MB_SERVICE_NOTIFICATION	WindowsNTでは、イベントをユーザーにサービス通知する。カレントデスクトップにメッセージボックスを表示する。コンピュータにユーザーがログオンしていなくても同様。このフラグがセットされると、*hWnd*パラメータはNULLでなければならない。これにより、メッセージボックスは*hWnd*に対応するデスクトップと別のデスクトップに表示できる。
MB_SETFOREGROUND	メッセージボックスが前面のウィンドウになる。Windowsは、メッセージボックスのためSetForeground Window関数を呼ぶ。
MB_SYSTEMMODAL	すべてのアプリケーションは、ユーザーがメッセージボックスに応答するまで中断される。アプリケーションがMB_ICONHANDを指定しなければ、メッセージボックスは、作成されてからモーダルになる。このため、親ウィンドウやそのほかのウィンドウは、メッセージボックスのアクティブ化によるメッセージを受け取り続ける。システムモーダルメッセージボックスは、すぐに対応を求められるような、重大で潜在的な危険性のあるエラー(たとえば、メモリ不足など)をユーザーに通知するために使われる。

MB_TASKMODAL	*hWnd*パラメータがNULLのとき、現在のタスクに属するすべてのトップレベルのウィンドウが使用不能になることを除いて、MB_APPLMODALと同じ。呼び出し側のアプリケーションやライブラリが使用可能なウィンドウのハンドルを持たず、ほかのアプリケーションを中断せずに現在のアプリケーションのほかのウィンドウへの入力を阻止しなければならないときには、このフラグを使用しなければならない。
MB_YESNO	メッセージボックスには、[はい]と[いいえ]の2つのプッシュボタンが含まれる。
MB_YESNOCANCEL	メッセージボックスには、[はい]、[いいえ]、[キャンセル]の3つのプッシュボタンが含まれる。

戻り値

メッセージボックスを作成するためのメモリが不足している場合は、0を返す。それ以外の場合は、ダイアログボックスが返した以下のメニュー項目値のいずれかを返す。

値	意味
IDABORT	[中止]ボタンが押された。
IDCANCEL	[キャンセル]ボタンが押された。
IDIGNORE	[無視]ボタンが押された。
IDNO	[いいえ]ボタンが押された。
IDOK	[OK]ボタンが押された。
IDRETRY	[再試行]ボタンが押された。
IDYES	[はい]ボタンが押された。

メッセージボックスに[キャンセル]ボタンがあるとき、Escキーまたは[キャンセル]ボタンが押されると、IDCANCEL値を返す。メッセージボックスに[キャンセル]ボタンがなければ、Escキーを押しても効果はない。

解　説

MessageBox関数は、メッセージボックスウィンドウの作成、表示、操作を行う。メッセージボックスには、アプリケーション定義のメッセージとタイトルや、*uType*パラメータが示す定義済みのアイコンとプッシュボタンの組み合わせが含まれる。システム用のメモリが少ないことを示すためのシステムモーダルメッセージボックスを作成するときは、*lpText*パラメータと*lpCaption*パラメータが指す文字列をリソースファイルから取得してはならない。このような状況でリソースにアクセスすると、リソースのロードに失敗する可能性がある。

アプリケーションがMB_ICONHANDフラグとMB_SYSTEMMODALフラグを*uType*パラメータに指定してMessageBox関数を呼び出すと、Windowsは、使用可能なメモリ容量にかかわらず、メッセージボックスをそのまま表示する。これらのフラグが指定されると、Windowsはメッセージボックスのテキストの長さを3行に制限する。Windowsはメッセージボックスにあわせて自動的に行を折り返すわけではないため、メッセージ文字列の適当な所に、行を折り返すための改行を入れなければならない。

ダイアログボックスが存在するときにメッセージボックスを作成するときは、ダイア

ログボックスのハンドルを*hWnd*パラメータとして使用する。*hWnd*パラメータでダイアログボックス内のコントロールなどの子ウィンドウを識別することはできない。Windows 95では、最大16,364個のウィンドウハンドルをサポートする。

参　照　FlashWindow,MessageBeep,MessageBoxEx,SetForegroundWindow

SendDlgItemMessage	user32.dll	95	NT
メッセージ送信			

V　C

```
LONG SendDlgItemMessage(
        HWND    hDlg,           //ダイアログボックスのハンドル
        int     nIDDlgItem,     //コントロールの識別子
        UINT    uMsg,           //送信メッセージ
        WPARAM  wParam,         //第1メッセージパラメータ
        LPARAM  lParam          //第2メッセージパラメータ
        );
```

V　B

```
Declare Function SendDlgItemMessage Lib "user32" Alias "SendDlgItemMessageA" (ByVal hDlg As Long, ByVal nIDDlg-Item As Long, ByVal wMsg As Long, ByVal wParam As Long, ByVal lParam As Long) As Long
```

パラメータ

- *hDlg*　コントロールを含むダイアログボックスの識別子。
- *nIDDlgItem*　メッセージを受け取るコントロールの識別子を指定する。
- *uMsg*　送信メッセージを指定する。
- *wParam*　メッセージにより異なる追加情報を指定する。
- *lParam*　メッセージにより異なる追加情報を指定する。

戻り値　戻り値は、メッセージ処理の結果を示す。結果の値は送られたメッセージにより異なる。

解　説　SendDlgItemMessage関数は、ダイアログボックス内のコントロールにメッセージを送る。

SendDlgItemMessage関数は、メッセージが処理されるまで制御を戻さない。

SendDlgItemMessage関数の動作は、指定されたコントロールのハンドルを取得してSendMessage関数を呼び出すことと同じである。

参　照　SendMessage

SetDlgItemInt	user32.dll	95	NT
コントロールテキスト設定			

V C

```
BOOL SetDlgItemInt(
    HWND   hDlg,          //ダイアログボックスのハンドル
    int    nIDDlgItem,    //コントロールの識別子
    UINT   uValue,        //設定する値
    BOOL   bSigned        //符号の有無を示すインジケータ
);
```

V B

```
Declare Function SetDlgItemInt Lib "user32" Alias "SetDlgItemInt" (ByVal hDlg As Long, ByVal nIDDlgItem As Long, ByVal wValue As Long, ByVal bSigned As Long) As Long
```

パラメータ

- *hDlg*　コントロールを含むダイアログボックスの識別子。
- *nIDDlgItem*　テキストを変更するコントロールを指定する。
- *uValue*　項目テキストを生成するための整数値を指定する。
- *bSigned*　*uValue*パラメータの値が符号付きまたは符号なしのどちらであるかを指定する。このパラメータがTRUEのときは、*uValue*は符号付きの値になる。このパラメータがTRUEで*uValue*が0未満の値のときは、文字列中の最初の数字の前にマイナス記号が付けられる。パラメータがFALSEのときは、*uValue*が符号なしの値になる。

戻り値

正常終了	TRUE
異常終了	FALSE

詳細なエラー情報はGetLastError関数で取得。

解　説

SetDlgItemInt関数は、ダイアログボックス内の指定されたコントロールのテキストを、対応する整数値で表された文字列に設定する。この関数は、指定されたコントロールにWM_SETTEXTメッセージを送る。

参　照

GetDlgItemInt,→SetDlgItemText,→WM_SETTEXT

SetDlgItemText	user32.dll	95	NT
コントロールテキスト設定			

V C

```
BOOL SetDlgItemText(
    HWND        hDlg,         //ダイアログボックスのハンドル
    int         nIDDlgItem,   //コントロールの識別子
    LPCTSTR     lpString      //設定するテキスト
    );
```

V B

```
Declare Function SetDlgItemText Lib "user32" Alias "SetDlgItemTextA" (ByVal hDlg As Long, ByVal nIDDlgItem As Long, ByVal lpString As String) As Long
```

パラメータ

- *hDlg*　コントロールを含むダイアログボックスの識別子。
- *nIDDlgItem*　テキストを設定するコントロールの識別子。
- *lpString*　コントロールにコピーするテキストを設定する、NULLで終わる文字列へのポインタ。

戻り値

正常終了	TRUE
異常終了	FALSE

詳細なエラー情報はGetLastError関数で取得。

解　説

SetDlgItemText関数は、ダイアログボックス内のコントロールのタイトルやテキストを設定する。この関数は、指定されたコントロールにWM_SETTEXTメッセージを送る。

参　照

GetDlgItemInt,GetDlgItemText,→SetDlgItemInt,→WM_SETTEXT

MessageBoxIndirect	user32.dll	95	
メッセージボックス作成			

V C

```
BOOL MessageBoxIndirect(
    LPMSGBOXPARAMS    lpMsgBoxParams //メッセージボックスパラメータのアドレス
    );
```

V B

Declare Function MessageBoxIndirect Lib "user32" Alias "MessageBoxIndirectA" (lpMsgBoxParams As MSGBOXPARAMS) As Long

パラメータ

・*lpMsgBoxParams*　メッセージボックスに表示する情報を含むMSGBOXPARAMS構造体のポインタ。

解　説

MessageBoxIndirect関数は、メッセージボックスの作成、表示、制御を行う。このメッセージボックスには、アプリケーション定義のメッセージテキストやタイトル、アイコン、ボタンの組み合わせが含まれる。Windows 95では、最大16,364個のウィンドウハンドルがサポートされる。

参　照

→MSGBOXPARAMS

Edit Control

EditWordBreakProc	――――――――	95	NT
エディットコントロールテキスト編集			

V　C

```
int CALLBACK EditWordBreakProc(
        LPTSTR    lpch,        //編集するテキストのポインタ
        int       ichCurrent,  //開始位置
        int       cch,         //編集するテキスト長
        int       code         //取得したアクション
        );
```

パラメータ

・*lpch*　編集コントロールのテキストのポインタ。

・*ichCurrent*　関数がワードを分けるためにチェックを開始するポインタ識別子のテキストバッファの文字位置を指定する。

・*cch*　編集コントロールテキストの文字列長を指定する。

・*code*　コールバック関数により取得されるアクションを指定する。このパラメータは、以下の値の1つである。

値	アクション
WB_ISDELIMITER	カレント位置の文字列の限界を検出。
WB_LEFT	カレント位置の左のワードの開始を検出。
WB_RIGHT	カレント位置の右のワードの開始を検出(これは、編集コントロールが右詰めであるときに有効)。

Windows 95では、豊富な編集コントロールを定義し、*code*パラメータに値を加えている。値のリストについては、EM_FINDWORDBREAKの記述を照会のこと。

戻り値

*code*パラメータがWB_ISDELIMITERを指定している場合、カレント位置が限界位置であれば戻り値は0以外(TRUE)で、そうでない場合は、0である。その他の場合は、戻り値はテキストバッファのワードの開始インデックスである。

解　説

EditWordBreakProc関数は、アプリケーション定義のコールバック関数である。EDITWORDBREAKPROCのタイプの値は、関数のポインタである。ラインフィードに続くキャリッジリターンは、コールバック関数により単一ワードとされる。2つのキャリッジリターンは単一ワードとしてラインフィードに続く。アプリケーションは、EM_SETWORDBREAKPROCメッセージでコールバック関数のアドレスを指定されているコールバック関数をインストールしなければならない。

EditWordBreakProcは、アプリケーション定義の関数名プレースホルダーである。

参　照

SendMessage,→EM_FINDWORDBREAK,→EM_SETWORDBREAKPROC

List Box

DlgDirList	user32.dll	95	NT
ファイル名取得			

V C

```
int DlgDirList(
        HWND     hDlg,            //リストボックスを持つダイアログボックスの
                                   ハンドル
        LPTSTR   lpPathSpec,      //パス名またはファイル名の文字列のポインタ
        int      nIDListBox,      //リストボックスの識別子
        int      nIDStaticPath,   //スタティックコントロールの識別子
        UINT     uFileType        //表示するファイルの属性
        );
```

V B

Declare Function DlgDirList Lib "user32" Alias "DlgDirListA" (ByVal hDlg As Long, ByVal lpPathSpec As String, ByVal nIDListBox As Long, ByVal nIDStaticPath As Long, ByVal wFileType As Long) As Long

パラメータ

- *hDlg*　リストボックスを持つダイアログボックスの識別子。
- *lpPathSpec*　パス名またはファイル名を含む、NULLで終わる文字列を指すポインタ。DlgDirListはこの文字列を変更することがあるため、文字列の領域は変更分を格納するのに十分な大きさでなければならない。詳しくは、解説を参照のこと。
- *nIDListBox*　リストボックスの識別子を指定する。このパラメータが0のとき、DlgDirListリストボックスが存在しないものとし、ファイル名やディレクトリを格納しない。
- *nIDStaticPath*　カレントドライブとディレクトリの表示のために使われるスタティックコントロールの識別子を指定する。このパラメータが0のとき、DlgDirListはそのようなコントロールが存在しないものとする。

・*uFileType*　　表示するファイル名の属性を指定する。このパラメータは、以下の値の組み合わせになる。

値	説明
DDL_ARCHIVE	アーカイブファイル。
DDL_DIRECTORY	サブディレクトリ。サブディレクトリは角括弧([])で括られる。
DDL_DRIVES	ドライブ。ドライブは[-x-]の形式でリストされる。xは、ドライブ文字である。
DDL_EXCLUSIVE	排他ビット。この排他ビットをセットすると、指定されたタイプのファイルだけが表示される。そうでないときは、指定されたファイルだけでなく通常のファイルもすべて表示される。
DDL_HIDDEN	隠しファイル。
DDL_READWRITE	ほかの属性を持たない読み書き可能なデータファイル。
DDL_READONLY	読み取り専用ファイル。
DDL_SYSTEM	システムファイル。
DDL_POSTMSGS	アプリケーションメッセージキューにポストメッセージを送る。デフォルトは、ダイアログボックスプロシージャに直接メッセージを送る。

戻り値

リストが作成された場合	0以外
検索パスに入力文字列を含む値がない場合	0

解説

DlgDirList関数は、指定されたパス名またはファイル名と一致するすべてのファイル名をリストボックスに格納する。

*lpPathSpec*パラメータに長さが0の文字列を指定したとき、またはファイル名を含まないディレクトリ名だけを指定したときには、文字列が*.*に変更される。

*lpPathSpec*パラメータの形式は以下の通り。

[drive:][[\u]directory[\idirectory]\u][filename]

driveはドライブ名、directoryは有効なディレクトリ名、filenameは少なくとも1つのワイルドカード(?or*)を含む有効なファイル名である。

*lpPathSpec*パラメータにドライブ名やディレクトリ名、またはその両方を指定したときは、リストボックスに格納する前にカレントドライブとカレントディレクトリが指定のドライブとディレクトリに変更される。*nIDStaticPath*パラメータの識別子のスタティックコントロールについても、ドライブ名やディレクトリ名が新しいものに更新される。

*lpPathSpec*パラメータのパス名とファイル名がリストボックスに格納されると、ドライブ名やディレクトリ名の部分が削除されて、*lpPathSpec*パラメータが更新される。

DlgDirList関数は、LB_RESETCONTENTメッセージとLB_DIRメッセージをリストボックスに送る。

参照

→DlgDirListComboBox,DlgDirSelectComboBoxEx,→DlgDirSelectEx

DlgDirSelectEx	user32.dll	95	NT
リストボックス選択項目取得			

V C

```
BOOL DlgDirSelectEx(
    HWND      hDlg,         //リストボックスを持つダイアログボックスのハンドル
    LPTSTR    lpString,     //パス名の文字列バッファへのポインタ
    int       nCount,       //パス名の文字列のバイト数
    int       nIDListBox//リストボックスの識別子
    );
```

V B

Declare Function DlgDirSelectEx Lib "user32" Alias "DlgDirSelectExA" (ByVal hWndDlg As Long, ByVal lpszPath As String, ByVal cbPath As Long, ByVal idListBox As Long) As Long

パラメータ

- *hDlg*　リストボックスを持つダイアログボックスの識別子。
- *lpString*　選択されたパス名またはファイル名を受け取るバッファへのポインタ。
- *nCount*　*lpString*パラメータが指すパス名またはファイル名の長さ(バイト単位)を指定する。
- *nIDListBox*　ダイアログボックス内のリストボックスの識別子を表す整数を指定する。

戻り値

選択項目がディレクトリ名	0以外
それ以外	0

詳細エラー情報は、GetLastError関数で取得。

解　説

DlgDirSelectEx関数は、単一選択のリストボックスのカレントで選択されている項目を取得する。指定されたリストボックスには、DlgDirList関数を使用してデータを格納しておく。選択項目は、ドライブ名、ファイル名、またはディレクトリ名として処理される。カレントに選択されている項目がディレクトリ名またはドライブ名のとき、DlgDirSelectBoxExは、それらの名前から角括弧やハイフンを取り除いて、新しいパス名に挿入できるようにする。何も選択されていないときには、*lpString*パラメータの内容は変わらない。

DlgDirSelectEx関数は、LB_GETCURSELメッセージとLB_GETTEXTメッセージをリストボックスに送る。

DlgDirSelectEx関数を使用して、リストボックスから1つ以上のファイル名を返すことはできない。複数選択のリストボックスではこの関数は使えない。使用した場合、この関数は0を返さず、*lpString*パラメータの指す文字列は変更されない。

参　照

→DlgDirList,→DlgDirListComboBox,DlgDirSelectComboBoxEx,→LB_GETCURSEL,→LB_GETTEXT

Scroll Bar

EnableScrollBar

EnableScrollBar	user32.dll	95	NT

スクロールバー状態設定

V　C

```
BOOL EnableScrollBar(
    HWND    hWnd,      //ウィンドウまたはスクロールバーのハンドル
    UINT    wSBflags,  //スクロールバータイプのフラグ
    UINT    wArrows    //スクロールバー矢印フラグ
    );
```

V　B

```
Declare Function EnableScrollBar Lib "user32" Alias "EnableScrollBar" (ByVal hwnd As Long, ByVal wSBflags As Long, ByVal wArrows As Long) As Long
```

パラメータ

・*hWnd*　*wSBflags*パラメータの値に応じた、ウィンドウまたはスクロールバーコントロールの識別子。

・*wSBflags*　スクロールバーのタイプを指定する。このパラメータは、次の値のいずれかになる。

値	意味
SB_BOTH	指定されたウィンドウの水平および垂直スクロールバーの矢印を使用可能または使用不能にする。*hwnd*パラメータはウィンドウのハンドルになる。
SB_CTL	スクロールバーをスクロールバーコントロールとして識別する。*hwnd*パラメータはスクロールバーコントロールのハンドルになる。
SB_HORZ	指定されたウィンドウの水平スクロールバーの矢印を、使用可能または使用不能にする。*hwnd*パラメータはウィンドウのハンドルになる。
SB_VERT	指定されたウィンドウの垂直スクロールバーの矢印を、使用可能または使用不能にする。*hwnd*パラメータはウィンドウのハンドルになる。

・*wArrows*　スクロールバーの矢印の使用可能または使用不能、および使用可能または使用不能にする矢印を指定する。このパラメータは、次の値のいずれかになる。

値	意味
ESB_DISABLE_BOTH	スクロールバーの両方の矢印を使用不能にする。
ESB_DISABLE_DOWN	垂直スクロールバーの下矢印を使用不能にする。
ESB_DISABLE_LEFT	水平スクロールバーの左矢印を使用不能にする。
ESB_DISABLE_LTUP	水平スクロールバーの左矢印か、垂直スクロールバーの上矢印を使用不能にする。
ESB_DISABLE_RIGHT	水平スクロールバーの右矢印を使用不能にする。
ESB_DISABLE_RTDN	水平スクロールバーの右矢印か、垂直スクロールバーの下矢印を使用不能にする。
ESB_DISABLE_UP	垂直スクロールバーの上矢印を使用不能にする。
ESB_ENABLE_BOTH	スクロールバーの両方の矢印を使用可能にする。

戻り値

指定通りに状態が変更された場合	TRUE
すでに要求状態もしくはエラー	FALSE

解説

EnableScrollBar関数は、スクロールバーの矢印の片方または両方を、使用可能または使用不能にする。

参照

→ShowScrollBar

GetScrollPos	user32.dll	95	NT
スクロールボックス位置取得			

VC

```
int GetScrollPos(
        HWND    hWnd,    //スクロールバー付きのハンドルのウィンドウ
        int     nBar     //スクロールバーフラグ
        );
```

VB

```
Declare Function GetScrollPos Lib "user32" Alias "GetScrollPos" (ByVal hwnd As Long, ByVal nBar As Long) As Long
```

パラメータ

・*hWnd* スクロールバーコントロールまたは標準スクロールバーを持つウィンドウの識別子。これは、*nBar*パラメータの値に依存する。

・*nBar* 調査するスクロールバーを指定する。以下の値のいずれかになる。

値	意味
SB_CTL	スクロールバーコントロールの位置を取得する。*hwnd*パラメータはスクロールバーコントロールのウィンドウハンドルになる。
SB_HORZ	ウィンドウの水平スクロールバーの位置を取得する。
SB_VERT	ウィンドウの垂直スクロールバーの位置を取得する。

戻り値

正常終了	スクロールボックスのカレント位置
異常終了	0

詳細なエラー情報は、GetLastError関数で取得。

解　説

GetScrollPos関数はスクロールバーのスクロールボックス(つまみ)のカレント位置を取得する。カレント位置は、カレントスクロール範囲に依存する相対的な値になる。たとえば、スクロール範囲が0から100であり、スクロールボックスがバーの中央にあるときは、カレント位置は50となる。

GetScrollPos関数は、32ビットスクロール位置をアプリケーションが使えるようにする。WM_HSCROLLやWM_VSCROLLメッセージのように、スクロールバーの位置を示すものは、位置データの16ビットが限界である。SetScrollPos、SetScrollRange、GetScrollPos、GetScrollRange関数は32ビットのスクロールバーの位置データをサポートする。それで、アプリケーションはWM_HSCROLLやWM_VSCROLLメッセージの処理中に32ビットスクロールバー位置データを得て、GetScrollPosを呼ぶことができる。

このテクニックが利用できるのは、ウィンドウコントロールでのスクロールの実時間内である。アプリケーションがWM_HSCROLLやWM_VSCROLLメッセージを処理することにより、スクロールボックスの位置を追ったり、ユーザーが動かす間つまみに知らせる、SB_THUMBTRACK通知メッセージを伝え、スクロールを行う。ユーザーが動かす間に、つまみの32ビット位置を検出する関数はない。GetScrollPosはスタティック位置データのみ用意する。アプリケーションはスクロールが場所を得た前後では、32ビット位置データを取得することだけできる。

参　照

→GetScrollRange,→ScrollDC,→ScrollWindow,→SetScrollPos,→SetScrollRange,
→WM_HSCROLL,→WM_VSCROLL

GetScrollRange | user32.dll | 95 | NT

スクロールバー最小値最大値取得

V　C

```
BOOL GetScrollRange(
    HWND    hWnd,       //スクロールバー付きのウィンドウのハンドル
    int     nBar,       //スクロールバーフラグ
    LPINT   lpMinPos,   //位置の最小値の取得アドレス
    LPINT   lpMaxPos    //位置の最高値を取得アドレス
    );
```

V　B

Declare Function GetScrollRange Lib "user32" Alias "GetScrollRange" (ByVal hwnd As Long, ByVal nBar As Long, lpMinPos As Long, lpMaxPos As Long) As Long

パラメータ

・*hWnd*　スクロールバーコントロールまたは標準スクロールバーを持つウィンドウの識別子。これは、*nBar*パラメータの値に依存する。

・*nBar*　どのスクロールバーを取得するかを指定する。このパラメータは以下の値のいずれかになる。

値	意味
SB_CTL	スクロールバーコントロールの範囲を取得する。*hwnd*パラメータはスクロールバーコントロールのハンドルである。
SB_HORZ	ウィンドウの水平スクロールバーを取得する。
SB_VERT	ウィンドウの垂直スクロールバーを取得する。

・*lpMinPos*：位置の最小値を受け取る整数値の変数を指すポインタ。
・*lpMaxPos*：位置の最大値を受け取る整数値の変数を指すポインタ。

戻り値

正常終了	TRUE
異常終了	FALSE

詳細なエラー情報は、GetLastError関数で取得。

解説

GetScrollRange関数は、指定されたスクロールバーのカレントスクロールバー位置の最小値と最大値を取得する。指定されたウィンドウが、標準スクロールバーを持たないときやスクロールバーコントロールでないときには、GetScrollRange関数は*lpMinPos*パラメータと*lpMaxPos*パラメータに0をコピーする。標準スクロールバーのデフォルトの範囲は0から100までである。スクロールバーコントロールのデフォルトの範囲は空(両方の値が0)である。
WM_HSCROLLやWM_VSCROLLメッセージのように、スクロールバーの位置を示すものは、位置データの16ビットが限界である。SetScrollPos、SetScrollRange、GetScrollPos、GetScrollRange関数は32ビットのスクロールバーの位置データをサポートする。WM_HSCROLLやWM_VSCROLLメッセージのため、16ビットの障害を逃れる方法はある。そのテクニックの説明は、GetScrollPos関数を照会のこと。

参照

→GetScrollPos,→SetScrollPos,→SetScrollRange,→WM_HSCROLL,
→WM_VSCROLL

ScrollDC	user32.dll	95	NT
矩形領域スクロール			

V C

```
ScrollDC(
    HDC          hDC,          //デバイスコンテキストのハンドル
    int          dx,           //水平スクロール量
    int          dy,           //垂直スクロール量
    CONST RECT   *lprcScroll,  //スクロール矩形のアドレス
    CONST RECT   *lprcClip,    //クリッピング矩形のアドレス
    HRGN         hrgnUpdate,   //スクロールリージョンのハンドル
    LPRECT       lprcUpdate    //更新矩形の構造体のアドレス
    );
```

V B

```
Declare Function ScrollDC Lib "user32" Alias "ScrollDC" (ByVal hdc As Long, ByVal dx As Long, ByVal dy As Long, lprcScroll As RECT, lprcClip As RECT, ByVal hrgnUpdate As Long, lprcUpdate As RECT) As Long
```

パラメータ

- *hDC*　スクロールするビットを含むデバイスコンテキストの識別子。
- *dx*　水平方向のスクロール量を指定する。このパラメータは、左へのスクロール値は負の数で指定する。
- *dy*　垂直方向のスクロール量を指定する。このパラメータは、上へのスクロール値は負の数で指定する。
- *lprcScroll*　スクロールする矩形の座標を設定する、RECT構造体のポインタ。
- *lprcClip*　クリッピング矩形の座標を設定する、RECT構造体のポインタ。デバイスビットはクリッピング矩形に割り当てられる。この構造体には、*lprcScroll*パラメータが指す元の矩形よりもこの矩形が小さいときは、小さいほうの矩形の中だけでスクロールが行われる。
- *hrgnUpdate*　スクロール処理により画面上に現れるリージョンの識別子。ScrollDC関数はこのリージョンを定義するが、必ずしも矩形であるとは限らない。
- *lprcUpdate*　スクロールにより更新されたリージョンを囲む矩形の座標を受け取るRECT構造体のポインタ。このリージョンは、再描画が必要な最大の矩形領域である。この構造体に格納される値は、デバイスコンテキストのマッピングモードにかかわらず、つねにクライアント座標になる。アプリケーションは、必要なときに更新リージョンを使用してInvalidateRgn関数を呼び出すことができる。

戻り値

正常終了	TRUE
異常終了	FALSE

詳細なエラー情報は、GetLastError関数で取得。

解　説　ScrollDC関数は、矩形領域のビットを垂直方向および水平方向にスクロールする。*lprcUpdate*パラメータがNULLのとき、Windowsは更新矩形の座標を計算しない。*hrgnUpdate*パラメータと*lprcUpdate*パラメータがいずれもNULLのときには、Windowsは更新リージョンを計算しない。*hrgnUpdate*がNULLでなければ、Windowsは、スクロール処理により表に出るリージョンの有効なハンドル(ScrollDC関数で定義)が、*hrgnUpdate*パラメータに設定されているものと仮定する。ウィンドウのクライアント領域全体をスクロールする必要があるときはScrollWindow関数を使い、それ以外のときはScrollDC関数を使う。

参　照　InvalidateRgn,→RECT,→ScrollWindow

ScrollWindow	user32.dll	95	NT
クライアント領域スクロール			

V　C

```
BOOL ScrollWindow(
    HWND          hWnd,         //スクロールするウィンドウのハンドル
    int           XAmount,      //水平スクロール量
    int           YAmount,      //垂直スクロール量
    CONST RECT    *lpRect,      //スクロール矩形の構造体のアドレス
    CONST RECT    *lpClipRect   //クリッピング矩形の構造体のアドレス
 );
```

V　B

Declare Function ScrollWindow Lib "user32" Alias "ScrollWindow" (ByVal hWnd As Long, ByVal XAmount As Long, ByVal YAmount As Long, lpRect As RECT, lpClipRect As RECT) As Long

パラメータ

- *hWnd*　スクロールするクライアント領域のウィンドウの識別子。
- *XAmount*　水平スクロール量をデバイス単位で指定する。CS_OWNDCまたは、CS_CLASSDCスタイルでスクロールする場合は、このパラメータはデバイス単位より論理単位を使う。左にスクロールするときは、このパラメータに負の値を指定する。
- *YAmount*　垂直スクロール量をデバイス単位で指定する。CS_OWNDCまたは、CS_CLASSDCスタイルでスクロールする場合は、このパラメータはデバイス単位より論理単位を使う。上にスクロールするときは、このパラメータに負の値を指定する。
- *lpRect*　クライアント領域内のスクロールする部分を指定するRECT構造体のポインタ。このパラメータがNULLのときは、クライアント領域全体がスクロールされる。
- *lpClipRect*　スクロールするときのクリッピング矩形を指定する、RECT構造体のポインタ。スクロールされるビットはこの矩形内のビットだけで、矩形外のビットはスクロールされない。

戻り値

正常終了	TRUE
異常終了	FALSE

詳細なエラー情報は、GetLastError関数により取得すること。

解説

ScrollWindow関数は、ウィンドウのクライアント領域の内容をスクロールする。スクロール時にウィンドウ内にキャレットがあるときには、ScrollWindowはキャレットを自動的に非表示にして消去されないようにし、スクロールが終了した後でキャレットを復元する。キャレットの位置は適切に調整される。

ScrollWindow関数により表に出る領域は再描画されないが、ウィンドウの更新リージョンの中に含めらる。アプリケーションは、領域の再描画が必要であることを通知するWM_PAINTメッセージを後で受け取る。表に出る領域をスクロールと同時に再描画するには、ScrollWindow関数を呼び出した直後にUpdateWindow関数を呼び出す。

*lprcScroll*パラメータがNULLのとき、スクロールされたウィンドウの子ウィンドウの位置は、*XAmount*パラメータと*YAmount*パラメータの分だけ相対的に移動し、ウィンドウ内の無効な(描画されていない)領域も移動する。このときにはScrollWindowの実行が高速になる。

l*prcScroll*パラメータがNULLでないときには、子ウィンドウの位置は変更されず、ウィンドウの無効領域も移動しない。*lpRect*にNULLを指定しない場合には、更新の問題を避けるため、ScrollWindow関数を呼び出す前にUpdateWindow関数を呼び出してウィンドウを再描画する。

参照

→RECT,→ScrollDC,ScrollWindowEx,UpdateWindow

SetScrollPos	user32.dll	95	NT
スクロールボックス位置設定			

V C

```
int SetScrollPos(
        HWND        hWnd,     //スクロールバーを持つウィンドウのハンドル
        int         nBar,     //スクロールバーフラグ
        int         nPos,     //スクロールボックスの新しい位置
        BOOL        bRedraw   //再描画フラグ
        );
```

V B

```
Declare Function SetScrollPos Lib "user32" Alias "SetScrollPos" (ByVal hwnd As Long, ByVal nBar As Long, ByVal nPos As Long, ByVal bRedraw As Long) As Long
```

パラメータ

・*hWnd*　設定するスクロールバーを持つウィンドウの識別子。

・*nBar*　設定するスクロールバーを指定する。この値は、以下のいずれかになる。

値	意味
SB_CTL	スクロールバー内のスクロールボックスの位置を設定する。このとき、*hWnd*パラメータはスクロールバーコントロールのハンドルである。
SB_HORZ	ウィンドウの標準水平スクロールバーのスクロールボックスの位置を設定する。
SB_VERT	ウィンドウの標準垂直スクロールバーのスクロールボックスの位置を設定する。

・*nPos*　スクロールボックスの新しい位置を指定する。その位置は、スクロールの範囲内になければならない。詳細なスクロール範囲の情報は、SetScrollRange関数で取得する。

・*bRedraw*　スクロールボックスの新しい位置を反映してスクロールバーを再描画するかどうかを指定する。このパラメータがTRUEのときはスクロールバーが再描画され、FALSEのときは再描画されない。

戻り値

正常終了	スクロールボックスの以前の位置
異常終了	0

詳細なエラー情報は、GetLastError関数で取得する。

解説

SetScrollPos関数は、スクロールボックス(つまみ)のカレント位置を設定する。オプションが指定されているときは、新しい位置を反映してスクロールバーを再描画する。

SetScrollPos関数は、Windows3.x.のWin32ベースを適合条件とする。ほかの関数を続けて呼び出すことによりスクロールバーが再描画されるときは、つねに*bRedraw*パラメータにFALSEを設定しておくと便利である。

WM_HSCROLLやWM_VSCROLLメッセージのように、スクロールバーの位置を示すものは、位置データの16ビットが限界である。アプリケーションはSetScrollPos関数の*nPos*パラメータに最大65,535の値を持つ位置データのメッセージを当てにするだけである。

SetScrollPos、SetScrollRange、GetScrollPos、GetScrollRange関数は32ビットのスクロールバーの位置データをサポートする。WM_HSCROLLやWM_VSCROLLメッセージのため、16ビットの障害を逃れる方法はある。そのテクニックの説明は、GetScrollPos関数を照会のこと。

参照

→GetScrollPos,→GetScrollRange,→ScrollDC,→ScrollWindow,→SetScrollRange

SetScrollRange	user32.dll	95	NT
スクロールバー最小値最大値設定			

VC

```
BOOL SetScrollRange(
        HWND    hWnd,      //スクロールバーを持つウィンドウのハンドル
        int     nBar,      //スクロールバーフラグ
        int     nMinPos,   //スクロール位置の最小値
        int     nMaxPos,   //スクロール位置の最大値
        BOOL    bRedraw    //再描画フラグ
        );
```

VB

```
Declare Function SetScrollRange Lib "user32" Alias "SetScrollRange" (ByVal hwnd As Long, ByVal nBar As Long, ByVal nMinPos As Long, ByVal nMaxPos As Long, ByVal bRedraw As Long) As Long
```

パラメータ

・*hWnd*　ウィンドウまたはスクロールバーの識別子。このパラメータに指定する値は、*nBar*パラメータの値による。

・*nBar*　設定するスクロールバーを指定する。このパラメータの値は、以下のいずれかになる。

値	意味
SB_CTL	スクロールバーの範囲を設定する。このとき、*hWnd*パラメータはスクロールバーのハンドルでなければならない。
SB_HORZ	ウィンドウの水平スクロールバーの範囲を設定する。
SB_VERT	ウィンドウの垂直スクロールバーの範囲を設定する。

・*nMinPos* スクロール位置の最小値を指定する。
・*nMaxPos* スクロール位置の最大値を指定する。
・*bRedraw* 位置の変更を反映してスクロールバーを再描画するかどうかを指定する。このパラメータがTRUEのときはスクロールバーが再描画され、FALSEのときは再描画されない。

戻り値

正常終了	TRUE
異常終了	FALSE

詳細なエラー情報は、GetLastError関数で取得。

解説

SetScrollRange関数は、指定されたスクロールバーの最小位置と最大位置の値を設定する。また、標準スクロールバーを非表示にしたり、表示したりする。アプリケーションでスクロールバー通知メッセージを処理している間は、この関数を呼び出してスクロールバーを非表示にしてはならない。

SetScrollPos関数の直後にSetScrollRange関数を呼び出すときには、スクロールバーが2回描画されないように、SetScrollPosの*bRedraw*パラメータに0を設定する。標準スクロールバーのデフォルトの範囲は0～100である。スクロールバーコントロールのデフォルトの範囲は空である(*nMinPos*および*nMaxPos*の値は0)。*nMinPos*と*nMaxPos*の各パラメータに指定する値の差は、MAXLONGの値以下でなければならない。

WM_HSCROLLやWM_VSCROLLメッセージのように、スクロールバーの位置を示すものは、位置データの16ビットが限界である。アプリケーションはSetScrollPos関数の*nPos*パラメータに最大65,535の値を持つ位置データのメッセージを当てにするだけである。

SetScrollPos、SetScrollRange、GetScrollPos、GetScrollRange関数は32ビットのスクロールバーの位置データをサポートする。WM_HSCROLLやWM_VSCROLLメッセージのため、16ビットの障害を逃れる方法はある。その技術および限界の説明は、GetScrollPos関数を照会のこと。

参照

→GetScrollPos,→GetScrollRange,→SetScrollPos

ShowScrollBar	user32.dll	95	NT
スクロールバー表示状態設定			

V　C

```
BOOL ShowScrollBar(
    HWND        hWnd,     //スクロールバーを持つウィンドウのハンドル
    int         wBar,     //スクロールバーフラグ
    BOOL        bShow     //スクロールバー表示フラグ
    );
```

V　B

```
Declare Function ShowScrollBar Lib "user32" Alias "ShowScrollBar" (ByVal hwnd As Long, ByVal wBar As Long, ByVal bShow As Long) As Long
```

パラメータ

- *hWnd*　*wBar*パラメータの値により、スクロールバーのコントロールの識別子、または標準スクロールバーを持つウィンドウの識別子。
- *wBar*　表示または非表示のスクロールバーを指定する。このパラメータは、以下の値のいずれかになる。

値	意味
SB_BOTH	ウィンドウの標準の水平方向と垂直方向のスクロールバーを表示または非表示。
SB_CTL	スクロールバーを表示または非表示にする。*hwnd*パラメータは、スクロールバーコントロールのハンドルとする。
SB_HORZ	ウィンドウの標準水平スクロールバーを表示または非表示。
SB_VERT	ウィンドウの標準垂直スクロールバーを表示または非表示。

- *bShow*　スクロールバーを表示するか非表示にするかを指定する。このパラメータがTRUEの場合はスクロールバーが表示され、それ以外の場合はスクロールバーが非表示になる。

戻り値

正常終了	TRUE
異常終了	FALSE

詳細なエラー情報は、GetLastError関数で取得。

解　説

ShowScrollBar関数は、スクロールバーを表示したり非表示にしたりする。アプリケーションでスクロールバー通知メッセージを処理しているときに、この関数を呼び出してスクロールバーを非表示にしない。

参　照

→GetScrollPos,→GetScrollRange,→ScrollDC,→ScrollWindow,→SetScrollPos,
→SetScrollRange

2. 4 Shell Features
シェル

ファイルのインストール、スクリーンセイバーととシェル操作に関するAPI関数群が含まれる。特に、ファイルインストール関数に関しては、非常に重要な役割を担う部分である。また、各アプリケーションから別のアプリケーションを実行したり、作成された文書に対する操作などもする操作などもシェル関数を利用する。

Control Panel

CPlApplet	————————	95	NT
コントロールパネルアプリケーションエントリポインタ			

V C

```
LONG APIENTRY CPlApplet
    HWN      hwndCPl,    //コントロールパネルウィンドウのハンドル
    UINT     uMsg,       //メッセージ
    LONG     lParam1,    //第1メッセージパラメータ
    LONG     lParam2     //第2メッセージパラメータ
    );
```

パラメータ

- *hwndCPl*　コントロールアプリケーションのメインウィンドウの識別子。
- *uMsg*　コントロールパネルアプリケーションに送られるメッセージを指定する。
- *lParam1*　メッセージにより異なる追加情報を指定する。
- *lParam2*　メッセージにより異なる追加情報を指定する。

戻り値

戻り値はメッセージにより異なる。詳細については、個々のコントロールパネルメッセージの説明を照会すること。

解　説

CPlApplet関数は、コントロールパネルアプリケーションのエントリポイントとして供給されるライブラリ定義のコールバック関数である。
*hwndCPl*パラメータは、親ウィンドウのハンドルが必要なダイアログボックスまたはウィンドウに対して使用する。

File Instration Utility

GetFileVersionInfo

GetFileVersionInfo	version.dll	95	NT
バージョン情報取得			

V C

```
BOOL GetFileVersionInfo(
        LPTSTR    lptstrFilename, //ファイル名のバッファへのポインタ
        DWORD     dwHandle,       //無視される
        DWORD     dwLen,          //バッファのサイズ
        LPVOID    lpData          //ファイルバージョン情報を受け取るバッファへ
                                    のポインタ
);
```

V B

```
Declare Function GetFileVersionInfo Lib "version.dll" Alias "GetFileVersionInfoA"
(ByVal lptstrFilename As String, ByVal dwHandle As Long, ByVal dwLen As Long,
lpData As Any) As Long
```

パラメータ

- *lptstrFilename*　対象ファイル名を指定したNULLで終わる文字列のポインタ。
- *dwHandle*　このパラメータは、無視される。
- *dwLen*　*lpData*により指定されるバッファのサイズをバイト単位で指定する。GetFileVersionInfoSize関数は、ファイルバージョン情報のバイト単位のサイズを決定する。*lpData*により指定されるバッファが十分な大きさでないときは、ファイルバージョン情報はバッファのサイズに合わせて短くなる。
- *lpData*　ファイルバージョン情報を受け取るバッファのポインタ。この値は、後でVerQueryValue関数を呼び出すときに使うことができる。ファイルバージョン情報は、16ビット(Unicode)フォーマットである。

戻り値

正常終了	TRUE
異常終了	FALSE

詳細なエラー情報は、GetLastError関数により取得。

解　説

GetFileVersionInfo関数は、指定されたファイルに関するバージョン情報を返す。
他のファイルインストール関数同様、GetFileVersionInfoはWin32ファイル形式でのみ動作するが、16ビットWindowsファイル形式では動作しない。

参　照

→GetFileVersionInfoSize,→VerQueryValue,→VS_VERSION_INFO

GetFileVersionInfoSize	version.dll	95	NT
バージョン情報サイズ取得			

V C

```
DWORD GetFileVersionInfoSize(
    LPTSTR      lptstrFilename,    //ファイル名のバッファへのポインタ
    LPDWORD     lpdwHandle         //0が返る変数へのポインタ
    );
```

V B

Declare Function GetFileVersionInfoSize Lib "version.dll" Alias "GetFileVersionInfoSizeA" (ByVal lptstrFilename As String, lpdwHandle As Long) As Long

パラメータ

- *lptstrFilename* 対象ファイル名が格納されているNULLで終わる文字列のポインタ。
- *lpdwHandle* この関数で0にセットされる変数のポインタ。

戻り値

正常終了	ファイルのバージョン情報サイズ（バイト単位）
異常終了	0

詳細なエラー情報は、GetLastError関数により取得。

解　説

GetFileVersionInfoSize関数は、オペレーティングシステムが指定されたファイルのバージョン情報を取得できるかどうかを判定する。バージョン情報が利用できるとき、GetFileVersionInfoSizeは情報のサイズをバイト単位で返す。他のファイルインストール関数同様、GetFileVersionInfoSizeはWin32ファイル形式でのみ動作するが、16ビットWindowsファイル形式では動作しない。

GetFileVersionInfoSize関数は、GetFileVersionInfo関数呼び出す前に呼び出さなければならない。

参　照

→GetFileVersionInfo,→VerQueryValue

VerFindFile	version.dll	95	NT
インストール位置判定			

V C

```
DWORD VerFindFile(
    DWORD   dwFlags,        //関数の動作を条件付けるビットフラグ
    LPTSTR  szFileName,     //インストールするファイル名
    LPTSTR  szWinDir,       //Windowsディレクトリ名
    LPTSTR  szAppDir,       //インストールするファイルを指すディレクトリ名
    LPTSTR  szCurDir,       //インストールするファイルのカレントディレク
                              トリ名を受け取る
    PUINT   lpuCurDirLen,   //カレントディレクトリのサイズ
    LPTSTR  szDestDir,      //インストール先のディレクトリのパス名を受け
                              取る
    PUINT   lpuDestDirLen   // szDestDirのサイズ
);
```

V B

Declare Function VerFindFile Lib "version.dll" Alias "VerFindFileA" (ByVal uFlags As Long, ByVal szFileName As String, ByVal szWinDir As String, ByVal szAppDir As String, ByVal szCurDir As String, lpuCurDirLen As Long, ByVal szDestDir As String, lpuDestDi

パラメータ

・d*wFlags*　フラグのビットマスク。このパラメータの値には、以下の値がある。

フラグ	説明
VFFF_ISSHAREDFILE	ソースファイルは、複数のアプリケーションにより共有される可能性がある。アプリケーションは、コピーされるファイルの場所を決定するためこの情報を使う。そのほかの値はすべて将来の使用のために予約されている。

・s*zFileName*　インストールするファイル名へのポインタ。このファイル名はファイル名と拡張子だけを含み、パスは含まない。

・*szWinDir*　Windowsが実行している、または実行されるディレクトリ名のポインタ。この文字列は、GetWindowsDirectory関数により返される。

・*szAppDir*　インストールプログラムが一連のファイルをインストールするディレクトリ名へのポインタ。インストールプログラムがアプリケーションをインストールする場合、このディレクトリがアプリケーションの常駐ディレクトリになる。アプリケーションの作業ディレクトリも、ほかに指定をしない限りこのディレクトリになる。

・*szCurDir*　インストール済みのファイル(現バージョン)へのパスが格納されるバッファのポインタ。ファイルがインストールされていない場合、バッファの長さが0となる。要求がなくても、バッファの長さは_MAX_PATHバイト以上でなければならない。

・*lpuCurDirLen*　*szCurDir*のバッファの長さを指定する値へのポインタ。このポインタはNULLではならない。関数が制御を戻すとき、*lpuCurDirLen*には、*szCurDir*に返された終端のNULLを含むキャラクタ単位のパスデータのサイズが格納される。バッファが小さすぎてすべてのデータが格納できない場合でも、*lpuCurDirLen*の値はパスを含んだバッファのサイズとなる。

・*szDestDir*　VerFindFile関数が最適であると判断したインストール場所のパスが格納されるバッファのポインタ。このパスは0で終わる文字列である。バッファの長さは、少なくとも_MAX_PATHバイトでなければならない。

・*lpuDestDirLen*　*szDestDir*のバッファの長さの値のポインタ。このポインタは、NULLではならない。関数が制御を戻すとき、*lpuDestDirLen*には、*szDestDir*に返された終端のNULL文字を含むパスデータのキャラクタサイズが格納される。バッファが小さすぎてすべてのデータが格納できない場合、*lpuDestDirLen*の値はパスを入れると必要なバッファのサイズとなる。

戻り値

ファイルの状態を示すビットマスクを、以下の値のいずれかを1つ以上返す。

値	意味
VFF_CURNEDEST	最適であると判断したインストール先に、インストールされたカレントバージョンのファイルがない。
VFF_FILEINUSE	インストールされたカレントバージョンのファイルをWindowsが使用中である。このため、ファイルに上書きしたり、削除したりすることができない。
VFF_BUFFTOOSMALL	サイズが小さすぎるため対応する文字列を格納できないバッファが、少なくとも1つある。アプリケーションは、出力バッファをチェックし、サイズの足りないバッファを判定しなければならない。

ほかの値はすべて将来の使用のために予約されている。

解　説

VerFindFile関数は、ファイルをインストールするときに、システム内にあるそのファイルの別バージョンの有無に基づいて、インストールする位置を判定する。VerFindFile関数の戻り値は、後でVerInstallFile関数呼び出しの中で使われる。
他のファイルインストール関数同様、VerFindFileはWin32ファイル形式でのみ動作するが、16ビットWindowsファイル形式では動作しない。
VerFindFile関数は、OpenFile関数を使用して、指定されたファイルの複製を検索する。また、指定されたWindowsディレクトリからシステムディレクトリを判定するか、そのパスを探索する。
ファイルがこのアプリケーションに対してプライベートであること(VFFF_ISSHAREDFILEではないこと)を*dwflags*パラメータが示す場合、VerFindFile関数はアプリケーションのディレクトリがインストールに最適なディレクトリであると判断する。そうでない場合、システムが共有のWindowsを実行しているときは、Windowsディレクトリが最適なインストールディレクトリであると判断する。システムがプライベートのWindowsを実行しているときには、SYSTEMディレクトリが最適であると判断する。

参　照

→GetWindowsDirectory,OpenFile,→VerInstallFile

VerInstallFile	version.dll	95	NT
ファイルインストール			

V　C

```
DWORD VerInstallFile(
        DWORD   uFlags,          //関数の実行状態フラグ
        LPTSTR  szSrcFileName,   //インストール元ファイル名
        LPTSTR  szDestFileName,  //インストール先ファイル名
        LPTSTR  szSrcDir,        //インストール元ディレクトリ名
        LPTSTR  szDestDir,       //インストール先ディレクトリ名
        LPTSTR  szCurDir,        //インストールされるカレントディレクトリ名
        LPTSTR  szTmpFile,       //インストール中使用される一時ファイル名
        PUINT   lpuTmpFileLen    //szTmpFileパラメータの文字列サイズ
        );
```

V　B

```
Declare Function VerInstallFile Lib "version.dll" Alias " VerInstallFileA" (ByVal uFlags As Long, ByVal szSrcFileName As String, ByVal szDestFileName As String, ByVal szSrcDir As String, ByVal szDestDir As String, ByVal szCurDir As String, ByVal szTmpFile  As String, lpuTmpFileLen As Long) As Long
```

パラメータ

- *uFlags*　フラグのビットマスク。このパラメータは以下の値の1つである。

フラグ	説明
VIFF_FORCEINSTALL	バージョン番号が不適切な場合にもファイルをインストールする。関数は、インストールの間、物理的なエラーだけをチェックする。
VIFF_DONTDELETEOLD	以前にインストールされたファイルがインストール先のディレクトリにないときは、インストール済みのファイルを削除せずにファイルをインストールする。

ほかの値はすべて将来の使用のために予約されている。

- *szSrcFileName*　インストールするファイル名へのポインタ。このファイル名は、*szSrcDir*が指すディレクトリにあるファイル名で、ベースファイル名と拡張子だけを指定し、パスを含めない。
- *szDestFileName*　VerInstallFile関数がインストール時に新しいファイルに指定するファイル名へのポインタ。このファイル名は、*szSrcFilename*のディレクトリ内のファイル名と異なることもある。新しいファイル名にはベースファイル名と拡張子だけを指定し、パスを含めない。
- *szSrcDir*　新しいファイルのあるディレクトリ名を格納するバッファへのポインタ。
- *szDestDir*　新しいファイルがインストールされるディレクトリ名を格納するバッファへのポインタ。VerFindFile関数の*szDestDir*パラメータに、この値が返される。
- *szCurDir*　既存のファイル(現バージョン)があるディレクトリ名を格納するバッファへのポインタ。VerFindFile関数の*szCurDir*パラメータに、この値が返される。
- *szTmpFile*　VerInstallFile関数の最初の呼び出しで空にするバッファへのポインタ。関数は、インストール元のファイルの一時的な複製ファイル名をこのバッファに格納する。バッファの長さは、少なくとも_MAX_PATHバイト以上でなければならない。
- *lpuTmpFileLen*　*szTmpFile*パラメータのバッファの長さの値へのポインタ。このパラメータは、NULLであってはならない。関数が制御を戻すとき、*lpuTmpFileLen*には、*szTmpFile*に返されたファイル名データの終端のNULL文字を含むバイト単位のサイズが格納される。バッファが小さすぎてすべてのデータが格納できない場合、*lpuTmpFileLen*の値はそのデータを格納した場合のサイズが設定される。

戻り値

戻り値は、例外を示すビットマスクである。以下の値のいずれかを1つ以上を返す。

値	意味
VIF_TEMPFILE	新しいファイルの一時的な複製ファイルがインストール先のディレクトリに存在する。失敗の原因はほかのフラグにも反映される。
VIF_MISMATCH	新しいファイルと既存のファイルの属性が1つ以上異なる。このエラーは、VIFF_FORCEINSTALLフラグをセットして再度VerInstallFile関数を呼び出すことにより、無効にできる。
VIF_SRCOLD	インストールするファイルが既存のファイルよりも古い。このエラーは、VIFF_FORCEINSTALLフラグをセットして再度VerInstallFile関数を呼び出すことにより、無効にできる。
VIF_DIFFLANG	新しいファイルと既存のファイルの言語またはコードページ値が異なる。このエラーは、VIFF_FORCEINSTALLフラグをセットして再度VerInstallFile関数を呼び出すことにより、無効にできる。
VIF_DIFFCODEPG	現在実行中のWindowsのバージョンが表示できないコードページを、新しいファイルが要求している。このエラーは、VIFF_FORCEINSTALLフラグをセットして再度VerInstallFile関数を呼び出すことにより、無効にできる。
VIF_DIFFTYPE	新しいファイルが、既存のファイルとは異なるタイプ、サブタイプ、またはオペレーティングシステムを持つ。このエラーは、VIFF_FORCEINSTALLフラグをセットして再度VerInstallFile関数を呼び出すことにより、無効にできる。
VIF_WRITEPROT	既存のファイルが書き込み禁止である。このエラーはVIFF_FORCEINSTALLフラグをセットして再度VerInstallFile関数を呼び出すことにより、無効にできる。
VIF_FILEINUSE	Windowsが既存のファイルを現在使用しているため削除できない。
VIF_OUTOFSPACE	インストール先のドライブのディスク容量が足りないため、関数が一時ファイルを作成できない。
VIF_ACCESSVIOLATION	アクセス違反により、ファイルの読み込み、作成、削除、または名前の変更に失敗した。
VIF_SHARINGVIOLATION	共有違反により、ファイルの読み込み、作成、削除、または名前の変更に失敗した
VIF_CANNOTCREATE	関数が一時ファイルを作成できない。具体的なエラーはほかのフラグにより示される。
VIF_CANNOTDELETE	インストール先のファイルや、ほかのディレクトリにある現バージョンのファイルを関数が削除できない。VIF_TEMPFILEビットがセットされている場合、インストール作業は失敗し、インストール先のファイルを削除できないことがある。

VIF_CANNOTDELETECUR	既存バージョンのファイルが削除されていない。VIFF_DONTDELETEOLDが指定されていない。
VIF_CANNOTRENAME	関数は一時ファイルの名前を変更できなかったにもかかわらず、インストール先のファイルをすでに削除した。
VIF_OUTOFMEMORY	メモリ不足のため、関数が要求された操作を完了できない。通常は、アプリケーションが圧縮ファイルを展開しようとしてメモリを使い切ったことを意味する。
VIF_CANNOTREADSRC	関数がインストール元のファイルを読み取れない。これは、不適切なパスが指定されたということを意味する。
VIF_CANNOTREADDST	関数がインストール先の(既存の)ファイルを読み取れない。このため、関数はファイルの属性を調べることができない。
VIF_BUFFTOOSMALL	*szTmpFile*バッファが小さすぎるため、一時ファイルの名前を格納できない。関数が制御を戻すとき、*lpuTmpFileLen*には、ファイル名の保持に必要なバッファのサイズが格納される。

ほかの値はすべて将来の使用のために予約されている。

解　説

VerInstallFile関数は、VerFindFile関数から返された情報に基づいて、ファイルのインストールを試みる。
VerInstallFile関数は圧縮ファイルを展開し、ユニークなファイル名を割り付け、日付の古いファイルなどのエラーをチェックする。
他のファイルインストール関数同様、VerInstallFileはWin32ファイル形式でのみ動作するが、16ビットWindowsファイル形式では動作しない。
VerInstallFile関数は、ファイルをインストールディレクトリからインストール先のディレクトリにコピーする。*szCurDir*に指定した以前バージョンのファイルが存在する場合、VerInstallFile関数はファイルのバージョン情報を比較する。既存のファイルのバージョンがインストールしようとしているファイルのバージョンよりも新しいとき、またはファイルの属性が全く異なるときには(たとえば、ファイルの言語属性が異なるとき)、1つ以上のエラー値を返す。
VerInstallFile関数は、一時ファイルをインストール先のディレクトリに残す。アプリケーションは、エラーを修復するか、一時ファイルを削除することができる。エラーを修復する場合、VerInstallFile関数は、既存のファイルを削除して、一時ファイルの名前をオリジナルファイル名に変更する。

参　照

→VerFindFile

VerLanguageName	kernel32.dll	95	NT
言語識別子変換			

V C

```
DWORD VerLanguageName(
        DWORD    wLang,   //Microsoft言語識別子
        LPTSTR   szLang,  //言語文字列のバッファへのポインタ
        DWORD    nSize    //バッファのサイズ
        );
```

V B

```
Declare Function VerLanguageName Lib "kernel32" Alias "VerLanguageNameA"
(ByVal wLang As Long, ByVal szLang As String, ByVal nSize As Long) As Long
```

パラメータ

・*wLang*　バイナリ形式のMicrosoft言語識別子を指定する。たとえば、VerLanguageNameは0x040Aをカスチリア系スペイン語に変換する。VerLanguageNameが識別子を認識できない場合、*szLang*パラメータにはデフォルトの文字列("Language-Neutral")を指すポインタが格納される。Win32がサポートする言語識別子について詳しくは、「解説」を参照。

・*szLang*　*wLang*パラメータで指定された言語を表すNULLで終わる文字列バッファのポインタ。

・*nSize*　*szLang*が指すバッファのサイズをバイト単位で指定する。

戻り値

正常終了	バッファに格納された文字列のバイト単位のサイズ。この値には文字列の終端のNULL文字は含まれない。この値がバッファサイズよりも大きい場合は、文字列の長さがバッファサイズに切り詰められる。
異常終了	不明な言語識別子(=0)は、エラーにならない。

解説

VerLanguageName関数は、バイナリ表現のMicrosoft言語識別子を、言語を表すテキストに変換する。
通常インストールプログラムは、VerQueryValue関数が返す言語識別子を変換するときにこの関数を使う。変換されたテキスト文字列は、言語の競合が発生した場合に、処理を続行する方法をユーザーにたずねるダイアログボックスの中で使われることがある。

以下は、使用できる言語識別子とその言語である。

識別子	言語	識別子	言語
0x0000	Language-Neutral(言語不明)	0x040E	ハンガリー語
0x0400	デフォルト言語	0x040F	アイスランド語
0x0401	アラビア語	0x0410	イタリア語
0x0402	ブルガリア語	0x0810	スイス系イタリア語
0x0403	カタロニア語	0x0411	日本語
0x0404	伝統的な中国語	0x0412	韓国語
0x0804	簡体字中国語	0x0413	オランダ語
0x0405	チェコ語	0x0813	ベルギー系オランダ語
0x0406	デンマーク語	0x0414	ノルウェー語-ブークモール
0x0407	ドイツ語	0x0814	ノルウェー語-ニーノシク
0x0807	スイス系ドイツ語	0x0415	ポーランド語
0x0408	ギリシャ語	0x0416	ブラジル系ポルトガル語
0x0409	アメリカ英語	0x0816	ポルトガル語
0x0809	イギリス英語	0x0417	レートロマンス語
0x0C09	オーストラリア英語	0x0418	ルーマニア語
0x1009	カナダ英語	0x0419	ロシア語
0x040A	カスチリア系スペイン語	0x041A	クロアチア-セルビア語(ラテン)
0x080A	メキシコ系スペイン語	0x081A	セルビア-クロアチア語(キリル)
0x0C0A	現代スペイン語	0x041B	スロバキア語
0x040B	フィンランド語	0x041C	アルバニア語
0x040C	フランス語	0x041D	スウェーデン語
0x080C	ベルギー系フランス語	0x041E	タイ語
0x0C0C	カナダ系フランス語	0x041F	トルコ語
0x100C	スイス系フランス語	0x0420	ウルドゥー語
0x040D	ヘブライ語	0x0421	バハサ語

参照

→VerQueryValue

VerQueryValue	version.dll	95	NT
バージョンリソース内容取得			

V　C

```
BOOL VerQueryValue(
    CONST LPVOID pBlock,      //バージョンリソースのバッファへのポインタ
    LPTSTR       lpSubBlock,  //取得する値へのポインタ
    LPVOID*      lplpBuffer,  //バージョンポインタのバッファへのポインタ
    PUINT        puLen        //バージョン値の長さを示すバッファへのポインタ
    );
```

V　B

```
Declare Function VerQueryValue Lib "version.dll" Alias "VerQueryValue" (pBlock As Any, ByVal lpSubBlock As String, ByVal lplpBuffer As Long, puLen As Long) As Long
```

パラメータ

・*pBlock*　GetFileVersionInfoが返すバージョン情報リソースを格納するバッファへのポインタ。

・*lpSubBlock*　取得するバージョン情報の値を指定する、0で終わる文字列へのポインタ。文字列には円記号(¥)で区切られた名前と、以下の形式のいずれかを指定できる。

形式	説明
¥	ルートブロックを指定する。関数は、バージョン情報リソース用のVS_FIXEDFILEINFO構造体へのポインタを取得する。
¥VarFileInfo¥Translation	変数情報ブロック内の変換テーブルを指定する。関数は、言語識別子および文字セット識別子の配列を指すポインタを取得する。アプリケーションはこれらの識別子を使用して、バージョン情報リソースの中に言語固有ブロックの名前を作成する。
¥StringFileInfo¥lang-charset¥string-name	言語固有ブロック内の値を指定する。lang-charsetの名前は、リソースの変換テーブルにある言語および文字セットの識別子のペアである。lang-charsetの名前は16進の文字列で指定しなければならない。string-nameの名前は、後述の「解説」で説明する定義済み文字列の中のいずれかになる。

・*lplpBuffer*　バージョン情報値のポインタを取得するバッファへのポインタ。

・*puLen*　バージョン情報値のバイト単位の長さを取得するバッファへのポインタ。

戻り値

バージョン情報が利用可能な場合	0以外
指定された名前が存在しない、または*lpvBlock*が指すリソースが無効の場合	0

解　説

VerQueryValue関数は、指定されたバージョン情報リソースから、選択されたバージョン情報を返す。適切なリソースを得るには、VerQueryValueを呼び出す前にGetFileVersionInfoを呼び出さなければならない。他のファイルインストール関数同様、VerQueryValueはWin32ファイル形式でのみ動作するが、16ビットWindowsファイル形式では動作しない。
Win32APIでは、以下のバージョン情報定義済み文字列を含む。

CompanyName、FileDescription、FileVersion、InternalName、
LegalCopyright、OriginalFilename、ProductName、ProductVersion

以下は、バージョン情報のブロックよりファイル定義の文字列値の取得方法である。

```
VerQueryValue(pBlock,
TEXT("\\StringFileInfo\\040904E4\\FileDescription"),
             &lpBuffer,
             &dwBytes);
```

VerQueryValue関数を呼び出す前にGetFileVersionInfoSizeやGetFileVersionInfo関数により、*pBlock*バッファを初期化する必要がある。

参　照

→GetFileVersionInfo,→GetFileVersionInfoSize,→VS_FIXEDFILEINFO,
→VS_VERSION_INFO

Screen Saver

DefScreenSaverProc	――――――	95	NT
スクリーンセーバーメッセージ処理			

V　C

```
LONG DefScreenSaverProc(
    HWND    hWnd,     //スクリーンセーバーウィンドウのハンドル
    UINT    msg,      //メッセージ
    WPARAM  wParam,   //第1メッセージパラメータ
    LPARAM  lParam    //第2メッセージパラメータ
    );
```

パラメータ

- *hWnd*　　スクリーンセーバーウィンドウの識別子。
- *msg*　　処理するメッセージを指定する。DefScreenSaverProc関数が応答するメッセージのうち、スクリーンセーバーの動作に関するものを「解説」に示す。これらのメッセージに対してスクリーンセーバーアプリケーションが異なる動作を実行しなければならない場合は、アプリケーションのScreenSaverProcウィンドウプロシージャがそのメッセージを処理する。

・*wParam*　　メッセージに依存する追加情報を指定する。
・*lParam*　　メッセージに依存する追加情報を指定する。

戻り値

戻り値はメッセージ処理の結果を示しており、送られたメッセージにより異なる。

解　説

DefScreenSaverProc関数は、スクリーンセーバーアプリケーションが処理しないメッセージのデフォルトの処理を行う。
スクリーンセーバーアプリケーションのScreenSaverProcウィンドウプロシージャは、DefWindowProc関数の代わりにDefScreenSaverProc関数を使わなければならない。
DefScreenSaverProc関数は、スクリーンセーバーの動作に影響を与えないメッセージをすべてDefWindowProcに渡す。
以下のテーブルにウィンドウメッセージ別のDefScreenSaverProcの処理を示す。

メッセージ	応答
WM_ACTIVATE,WM_ACTIVATEAPP	*wParam*がFALSEであれば、スクリーンセーバーをクローズする。
WM_NCACTIVATE	*wParam*の値がFALSEのときは、スクリーンセーバーが入力フォーカスを失おうとしていることを示す。スクリーンセーバーは、WM_CLOSEメッセージを送ることによりクローズする。
WM_SETCURSOR,WM_RBUTTON	カーソルをNULLに設定して、画面からカーソルを削除する。DOWN,WM_MBUTTON DOWN,WM_PostQuitMessag関数を呼び、スクリーンセーバーをクローズする。
WM_DESTROY	WM_CLOSEメッセージにより、スクリーンセーバーウィンドウをクローズする。
WM_SYSCOMMAND	WM_SYSCOMMANDメッセージの*wParam*パラメータがSC_CLOSEかSC_SCREENSAVEならば、FALSEを返す。

参　照

DefWindowProc,PostQuitMessage,→ScreenSaverProc,→WM_ACTIVATE,
→WM_ACTIVATEAPP,→WM_CLOSE,→WM_DESTROY,→WM_KEYDOWN,
→WM_KEYUP,→WM_LBUTTONDOWN,→WM_MBUTTONDOWN,
→WM_MOUSEMOVE,→WM_NCACTIVATE,→WM_RBUTTONDOWN,
→WM_SETCURSOR,→WM_SYSCOMMAND

RegisterDialogClasses	――――――	95	NT
ウィンドウクラス登録			

V C

```
BOOL RegisterDialogClasses(
    HANDLE          hInst    //アプリケーションインスタンスのハンドル
    );
```

パラメータ

・*hInst*：ウィンドウクラスを登録するモジュールのインスタンスの識別子。

戻り値

正常終了	TRUE
異常終了	FALSE

詳細なエラー情報については、GetLastError関数で取得。

解　説

RegisterDialogClasses関数は、スクリーンセーバーの設定用のダイアログボックスで必要な、標準以外のウィンドウクラスを登録する。
RegisterDialogClasses関数はエクスポートしてはならない。この関数は、"SCRNSAVE.LIB"ファイルで定義されているルーチンから呼び出される。
スクリーンセーバーが設定用のダイアログボックスに対して特別なウィンドウクラスを登録しないとき、RegisterDialogClasses関数はTRUEを返す。

参　照

→ScreenSaverConfigureDialog

ScreenSaverConfigureDialog	――――――	95	NT
スクリーンセーバー設定ダイアログメッセージ受け取り			

V C

```
BOOL ScreenSaverConfigureDialog(
    HWND    hDlg,     //ダイアログボックスのハンドル
    UINT    message,  //メッセージ
    WPARAM  wParam,   //第1メッセージパラメータ
    LPARAM  lParam    //第2メッセージパラメータ
    );
```

パラメータ

- *hDlg*　設定用のダイアログボックスの識別子。
- *message*　メッセージを指定する。
- *wParam*　メッセージに依存する追加情報を指定する。
- *lParam*　メッセージに依存する追加情報を指定する。

戻り値

正常終了	TRUE
異常終了	FALSE

ScreenSaverConfigureDialog関数は、WM_INITDIALOGメッセージに応答する場合、SetFocus関数を呼び出してダイアログボックスのコントロールのいずれかに入力フォーカスを設定するならばFALSEを返す。それ以外の場合はTRUEを返し、フォーカスが与えられるダイアログボックスの最初のコントロールにキーボードフォーカスが設定される。

解　説

ScreenSaverConfigureDialog関数は、スクリーンセーバーの設定用のダイアログボックスに送られたメッセージを受け取る。ユーザーによる設定をサポートするスクリーンセーバーは、この関数を利用しなければならない。
設定用のダイアログボックスのダイアログボックステンプレートには、DLG_SCRNSAVECONFIGURE識別子を指定しなければならない。
ダイアログボックスプロシージャは、ダイアログボックスにデフォルトのウィンドウクラス(WC_DIALOG)が使われている場合にのみ使われる。ダイアログボックステンプレートでクラスが明示的に指定されていない場合アプリケーションは、デフォルトのクラスを使う。ダイアログボックスプロシージャはウィンドウプロシージャに似ているが、不要なメッセージを処理するためにDefWindowProc関数を呼び出してはならない。不要なメッセージは、デフォルトのダイアログボックスプロシージャが内部的に処理する。
ScreenSaverConfigureDialog関数は、アプリケーションのモジュール定義ファイル(.DEF)のEXPORTSでエクスポートしなければならない。

参　照

DefWindowProc,RegisterDialogClasses,SetFocus

ScreenSaverProc	————————	95	NT
スクリーンセーバーメッセージ処理			

V C

```
LONG ScreenSaverProc(
    HWND     hWnd,     //スクリーンセーバーウィンドウのハンドル
    UINT     message,  //メッセージ
    WPARAM   wParam,   //第1メッセージパラメータ
    LPARAM   lParam    //第2メッセージパラメータ
    );
```

パラメータ

- *hWnd*　ウィンドウの識別子。
- *message*　メッセージを指定する。
- *wParam*　メッセージに依存する追加情報を指定する。
- *lParam*　メッセージに依存する追加情報を指定する。

戻り値

メッセージ処理の結果を返す。値は、処理されたメッセージにより異なる。

解　説

ScreenSaverProc関数は、スクリーンセーバーウィンドウに送られたメッセージを受け取る。

スクリーンセーバーのScreenSaverProcウィンドウプロシージャは、DefWindowProc関数の代わりにDefScreenSaverProc関数を使用してデフォルトのメッセージを処理しなければならない。DefScreenSaverProc関数は、スクリーンセーバーの動作に影響を与えないメッセージをDefWindowProc関数に渡す。

ScreenSaverProc関数は、アプリケーションのモジュール定義ファイル(.DEF)のEXPORTSでエクスポートしなければならない。

参　照

→DefScreenSaverProc,DefWindowProc

Shell Libraty

DragAcceptFiles	shell32.dll	95	NT
ドロップファイル受け入れ登録			

V C

```
void DragAcceptFiles(
        HWND        hWnd,       //登録するウィンドウのハンドル
        BOOL        fAccept     //承諾オプション
        );
```

V B

Declare Sub DragAcceptFiles Lib "shell32.dll" Alias "DragAcceptFiles" (ByVal hwnd As Long, ByVal fAccept As Long)

パラメータ

- *hWnd*　ドロップされたファイルを受け入れるかどうかを登録するウィンドウの識別子。
- *fAccept*　*hWnd*パラメータで指定されたウィンドウがドロップされたファイルを受け入れるかどうかを指定する。ドロップされたファイルを受け入れるときはこの値にTRUEを、ファイルを受け入れないようにするときはFALSEを指定する。

解　説

DragAcceptFiles関数は、指定されたウィンドウがドロップされたファイルを受け入れるかどうかを登録する。
アプリケーションが *fAccept*パラメータにTRUEを設定してDragAcceptFiles関数を呼び出した場合、ファイルマネージャから送られるWM_DROPFILESメッセージにより処理を識別する。

参　照

→WM_DROPFILES

DragFinish	shell32.dll	95	NT
ドロップファイルメモリ解放			

V C

```
void DragFinish(
        HDROP       hDrop       //解放するメモリのハンドル
        );
```

V B

Declare Sub DragFinish Lib "shell32.dll" Alias "DragFinish" (ByVal hDrop As Long)

パラメータ

- *hDrop*　ドロップされたファイルを記述する構造体の識別子。このハンドルは、WM_DROPFILESメッセージのwParamパラメータでアプリケーションに渡される。

解説 DragFinish関数は、Windowsがファイル名をアプリケーションに転送するために割り当てたメモリを解放する。

参照 →WM_DROPFILES

DragQueryFile	shell32.dll	95	NT
ドロップファイル名取得			

VC

```
UINT DragQueryFile(
    HDROP    hDrop,      //ドロップされたファイルの構造体のハンドル
    UINT     iFile,      //問い合わせるファイルのインデックス
    LPTSTR   lpszFile,   //返されるファイル名のバッファへのポインタ
    UINT     cch         //ファイル名のバッファのサイズ
    );
```

VB

```
Declare Function DragQueryFile Lib "shell32.dll" Alias "DragQueryFileA" (ByVal HDROP As Long, ByVal UINT As Long, ByVal lpStr As String, ByVal ch As Long) As Long
```

パラメータ

- *hDrop* ドロップされたファイルのファイル名を含む構造体の識別子。
- *iFile* 問い合わせるファイルのインデックスを指定する。*iFile*パラメータの値が0xFFFFFFFFのときには、DragQueryFile関数はドロップされたファイルの個数を返す。*iFile*パラメータの値が0からドロップされたファイルの総数の間のときには、その値に対応するファイル名を*lpszFile*パラメータが指すバッファにコピーする。
- *lpszFile* 関数は、ドロップされたファイルのファイル名をこのバッファに返す。このファイル名は、NULLで終わる文字列である。このパラメータがNULLのとき、DragQueryFile関数はファイル名のバッファに必要なサイズをバイト単位で返します。
- *cch* *lpszFile*バッファのサイズをバイト単位で指定する。

戻り値 関数がファイル名をバッファにコピーした場合は、コピーされたバイト数を返す。このとき終端のNULLは含まない。
*iFile*が0xFFFFFFFFの場合は、ドロップされたファイルの個数が返される。
*iFile*が0からドロップされたファイルの総数の間で、*lpszFile*がNULLの場合は、バッファに必要なサイズが返される。この時終端のNULLは含まない。

解説 DragQueryFile関数は、ドロップされたファイルのファイル名を取得する。

参照 →DragQueryPoint

DragQueryPoint	shell32.dll	95	NT
ファイルドロップ時のマウス座標取得			

V　C

```
BOOL DragQueryPoint(
    HDROP      hDrop,   //ドロップされたファイルの構造体のハンドル
    LPPOINT    lppt     //マウス座標の構造体へのポインタ
    );
```

V　B

```
Declare Function DragQueryPoint Lib "shell32.dll" Alias "DragQueryPoint" (ByVal HDROP As Long, lpPoint As POINTAPI) As Long
```

パラメータ

- *hDrop*　ドロップされたファイルを記述する構造体の識別子。
- *lppt*　関数がファイルのドロップされたときのマウス位置の座標を格納する、POINT構造体へのポインタ。

戻り値

ファイルがクライアント領域にドロップされた場合	TRUE
それ以外	FALSE

解　説

DragQueryPoint関数は、ユーザーがマウスの左ボタンを離したときのカーソル位置の座標をPOINT構造体に格納する。座標が返されるウィンドウは、WM_DROPFILESメッセージを受け取ったウィンドウである。DragQueryPoint関数は、ファイルがドロップされたときのマウスが指す座標を取得する。

参　照

→DragQueryFile,→POINT,→WM_DROPFILES

ExtractAssociatedIcon	shell32.dll	95	NT
アイコンハンドルの取得			

V C

```
HICON ExtractAssociatedIcon(
    HINSTANCE   hInst,        //アプリケーション手続きハンドル
    LPTSTR      lpIconPath,   //アイコンのパスとファイル名へのポインタ
    LPWORD      lpiIcon       //アイコンのインデックスへのポインタ
    );
```

V B

```
Declare Function ExtractAssociatedIcon Lib "shell32.dll" Alias "ExtractAssociateIconA"
(ByVal hInst As Long, ByVal lpIconPath As String, lpiIcon As Long) As Long
```

パラメータ

- *hInst*　アプリケーションがこの関数を呼ぶ手続きを指定する。
- *lpIconPath*　要求されたアイコンの完全なパス名とファイル名を指定する文字列へのポインタ。関数は指定のファイルから、または指定のファイルに関連する実行ファイルより、アイコンハンドルを引き出す。アイコンハンドルが実行ファイルより取得された場合、関数は、*lpIconPath*により指される文字列の中に、実行する完全なパス名とファイル名を格納する。
- *lpiIcon*　取得されたアイコンハンドルのインデックスを指定するWORDポインタ。実行ファイルから取得されたアイコンハンドルは、*lpIcon*により指されたWORDポインタの中にアイコンの識別子を格納する。

戻り値

正常終了	アイコンハンドル
異常終了	NULL

アイコンが関連する実行ファイルより取得された場合、関数は、*lpIconPath*で指される文字列の中に実行ファイルの完全なパス名とファイル名を格納する。また、*lpiIcon*により指されるWORDポインタの中にアイコンの識別子を格納する。
関数が正常に処理されない場合は、NULLを返す。

解　説

ExtractAssociatedIcon関数は、ファイルや関連する実行ファイルの中からアイコンを検出しそのハンドルを返す。
ExtractAssociatedIcon関数は、最初に*lpIconPath*で指定されたアイコンを探す。関数が、そのファイルよりアイコンハンドルを取り出せなかった場合、実行ファイルの中を探す。ファイル名の拡張子に基づき関連付けされた実行ファイルは、ユーザーごとに情報を登録され、ファイルの関連付け時の表示機能として利用が可能になっている。

参　照

→ExtractIcon

ExtractIcon	shell32.dll	95	NT
アイコンハンドル取得			

V　C

```
HICON ExtractIcon(
    HINSTANCE    hInst,             //インスタンスハンドル
    LPCTST       RlpszExeFileName,//アイコンファイル名
    UINT         nIconIndex         //取得するアイコンのインデックス
    );
```

V　B

Declare Function ExtractIcon Lib "shell32.dll" Alias "ExtractIconA" (ByVal hInst As Long, ByVal lpszExeFileName As String, ByVal nIconIndex As Long) As Long

パラメータ

- *hInst*　関数を呼び出すアプリケーションのインスタンスの識別子。
- *lpszExeFileName*　実行可能ファイル、DLL、またはアイコンファイルを指定するNULLで終わる文字列へのポインタ。
- *nIconIndex*　取得するアイコンのインデックスを指定する。この値が0のときは、指定されたファイルの最初のアイコンハンドルを返す。この値が-1のときは、指定されたファイルにあるアイコンの総数を返す。

戻り値

正常終了	アイコンハンドル
実行可能ファイル、DLLアイコンのいずれでもない場合	1
ファイルにアイコンが存在しない場合	NULL

解　説

ExtractIcon関数は、指定された実行可能ファイル、ダイナミックリンクライブラリ(DLL)、またはアイコンファイルから、アイコンのハンドルを取得する。

FindExecutable	shell32.dll	95	NT
関連する実行ファイル取得			

V C

```
HINSTANCE FindExecutable(
    LPCTSTR   lpFile,       //ファイル名の文字列のアドレス
    LPCTSTR   lpDirectory,  //デフォルトディレクトリの文字列へのポインタ
    LPTSTR    lpResult      //実行可能ファイルが返される文字列へのポインタ
);
```

V B

Declare Function FindExecutable Lib "shell32.dll" Alias "FindExecutableA" (ByVal lpFile As String, ByVal lpDirectory As String, ByVal lpResult As String) As Long

パラメータ

- *lpFile*　ファイル名を指定する、NULLで終わる文字列のポインタ。ここには、文書ファイルか実行可能ファイルを指定できる。
- *lpDirectory*　デフォルトディレクトリを指定するNULLで終わる文字列へのポインタ。
- *lpResult*　実行可能ファイルの名前が返さるバッファへのポインタ。このファイル名は、NULLで終わる文字列である。この文字列は、[開く]コマンドが選択されたときに起動されるファイルを示し、*lpFile*パラメータで指定されるファイルを実行する。

戻り値

FindExecutable関数の戻り値は、関数が失敗したときに有効である。関数が正常に終了しなかった場合の戻り値は32以下である。発生する可能性のあるエラー値は、「解説」に記述する。

解　説

FindExecutable関数は、指定されたファイル名に関連付けられている実行可能ファイル(.EXE)の名前やハンドルを取得する。
FindExecutable関数の戻り値は、*lpResult*パラメータがDDEの対話を初期化する要求に対しサーバが応答しないとき、開始されているDDEサーバのパス名を設定する。
指定されたファイルタイプに対する関連付けがない場合、FindExecutable関数は31を返す。以下は、そのほかに発生する可能性のあるエラー値である。

値	意味
0	システムメモリ、またはリソースが足りない。
ERROR_FILE_NOT_FOUND	指定されたファイルが存在しない。
ERROR_PATH_NOT_FOUND	指定されたパスが存在しない。
ERROR_BAD_FORMAT	.EXEファイルの不正(Win32の.EXEでない、または.EXE形式エラー)。

参　照

→ShellExecute

ShellAbout	shell32.dll	95	NT
Windowsが提供するバージョン情報のダイアログボックスを表示			

V　C

```
int ShellAbout(
    HWND        hWnd,           //親ウィンドウハンドル
    LPCTSTR     szApp,          //タイトルバーと1行目のテキスト
    LPCTSTR     szOtherStuff,   //他のダイアログテキスト
    HICON       hIcon           //表示するアイコン
    );
```

V　B

```
Declare Function ShellAbout Lib "shell32.dll" Alias "ShellAboutA" (ByVal hwnd As Long, ByVal szApp As String, ByVal szOtherStuff As String, ByVal hIcon As Long) As Long
```

パラメータ

- *hWnd*　親ウィンドウの識別子。このパラメータは、NULLでもよい。
- *szApp*　ダイアログボックスのタイトルバーに表示するテキストと"MicrosoftWindows"または"MicrosoftWindowsNT"の次の行に表示するテキストへのポインタ。テキストに"#"が含まれていると2つの部分に分れてしまい、タイトルバーに#より左の部分を、"MicrosoftWindows"または"MicrosoftWindowsNT"の次の行に#の右の部分を表示する。
- *szOtherStuff*　バージョンと著作権表示の下に表示するテキストへのポインタ。
- *hIcon*　関数がダイアログボックスに表示するアイコンの識別子。このパラメータがNULLの場合、関数は、MicrosoftWindowsまたはMicrosoftWindowsNTのアイコンを表示する。

戻り値

正常終了	TRUE
異常終了	FALSE

解　説

ShellAbout関数は、Windowsが提供するバージョン情報のダイアログボックスを表示する。

Windowsが提供するバージョン情報のダイアログボックスとは、[マイコンピュータ]の[ヘルプ]-[バージョン情報]で表示されるダイアログボックスである。

ShellExecute	shell32.dll	95	NT
指定ファイルのオープン、表示、実行			

V C

```
HINSTANCE ShellExecute(
  HWND     hwnd,          //親ウィンドウのハンドル
  LPCTSTR  lpOperation,   //実行する操作の文字列へのポインタ
  LPCTSTR  lpFile,        //ファイル名やフォルダ名の文字列へのポインタ
  LPCTSTR  lpParameters,  //実行可能ファイルのパラメータの文字列へのポイ
                            ンタ
  LPCTSTR  lpDirectory,   //デフォルトディレクトリの文字列へのポインタ
  INT      nShowCmd       //オープン時のファイルの表示の有無
  );
```

V B

```
Declare Function ShellExcute Lib "shell32.dll" Alias "ShellExcuteA" (ByVal hwnd As Long ByVal lpOperation As String, ByVal lpFile As String, ByVal lpParameters As String, ByVal lpDirectory As String, ByVal nShowCmd As Long) As Long
```

パラメータ

・*hwnd* 　親ウィンドウの識別子。このウィンドウは、アプリケーションが生成するあらゆるメッセージボックスを受け取る。たとえば、アプリケーションはメッセージボックスに、生成するエラーの報告をする。

・*lpOperation* 　実行する操作を指定する、NULLで終わる文字列のポインタ。以下が文字列の値である。

文字列	意味
"open"	関数が*lpFile*で指定されたファイルをオープンする。ファイルは、実行可能ファイルか文書ファイルである。Windows 95では、ファイルはWindows 95フォルダである。
"print"	関数が*lpFile*で指定されたファイルを印刷する。ファイルは、文書ファイルである。ファイルが実行可能ファイルのときは、"open"が指定されたときと同様、ファイルをオープンする。
"explore"	Windows 95のみ。関数が*lpFile*で指定されたフォルダを検索する。

・*lpFile* 　オープンまたは表示するファイルを指定する、NULLで終わる文字列へのポインタ。この関数は、実行可能ファイルまたは文書ファイルをオープンできる。また、文書ファイルの印刷もできる。Windows 95では、*lpFile*はオープンまたは検索するためのWindows 95フォルダを指定するNULLで終わる文字列へのポインタである。

・*lpParameters* 　NULLで終わる文字列へのポインタ。*lpFile*パラメータに実行可能ファイルを指定するときは、アプリケーションに渡すパラメータをここに指定する。*lpFile*パラメータに文書ファイルを指定するときは、このパラメータはNULLである。

・*lpDirectory*　　デフォルトのディレクトリを指定する、NULLで終わる文字列へのポインタ。

・*nShowCmd*　　*lpFile*に実行可能ファイルを指定した場合は、*nShowCmd*にはオープンするときのアプリケーションの表示方法を指定する。このパラメータは、以下の値のいずれかである。

値	意味
SW_HIDE	ウィンドウを非表示にし、ほかのウィンドウをアクティブ化する。
SW_MAXIMIZE	指定されたウィンドウを最大表示にする。
SW_MINIMIZE	指定されたウィンドウをアイコン化し、システムのタスクリスト内のトップレベルウィンドウをアクティブ化する。
SW_RESTORE	ウィンドウをアクティブ化し表示する。ウィンドウがアイコン化または最大化されているときには、元のサイズと位置に復元する。アプリケーションは、アイコン化されたウィンドウを復元するときこのフラグを指定する。
SW_SHOW	ウィンドウをアクティブ化し、カレントサイズと位置で表示する。
SW_SHOWDEFAULT	アプリケーションを開始したプログラムによりCreateProcess関数を通じてSTARTUPINFO構造体に指定されたSW_フラグで表示状態を設定する。アプリケーションは、メインウィンドウの初期表示状態をこのフラグに設定するためShowWindow関数を呼ぶべきである。
SW_SHOWMAXIMIZED	ウィンドウをアクティブ化し、最大化ウィンドウとして表示する。
SW_SHOWMINIMIZED	ウィンドウをアクティブ化し、アイコン表示にする。
SW_SHOWMINNOACTIVE	ウィンドウをアイコン表示にする。アクティブなウィンドウは、アクティブな状態のまま。
SW_SHOWNA	ウィンドウをカレントの状態で表示する。アクティブなウィンドウは、アクティブな状態のまま。
SW_SHOWNOACTIVATE	以前に表示されたサイズと位置でウィンドウを表示する。アクティブなウィンドウは、アクティブな状態のまま。
SW_SHOWNORMAL	ウィンドウをアクティブ化し表示する。ウィンドウがアイコン化または最大化されているときには、元のサイズと位置に復元する。アプリケーションは最初にウィンドウを表示するとき、このフラグを指定すべきである。*lpFile*が文書ファイルを指定するとき、*nShowCmd*は0である。

戻り値

関数が正常に終了した場合は、実行するアプリケーションのインスタンスハンドル、またはダイナミックデータエクスチェンジ(DDE)サーバアプリケーションのハンドルを返す。
関数が正常に終了しなかった場合は32以下の値を返す。以下は、そのエラー値である。

値	意味
0	オペレーティングシステムのメモリかリソースが足りない。
ERROR_FILE_NOT_FOUND	指定されたファイルが見つからない。
ERROR_PATH_NOT_FOUND	指定されたパスが見つからない。
ERROR_BAD_FORMAT	.EXEファイルの不正(Win32以外の.EXEまたは.EXE形式のエラー)。
SE_ERR_ACCESSDENIED	Windows 95のみ。オペレーティングシステムは、指定されたファイルへのアクセスを拒否した。
SE_ERR_ASSOCINCOMPLETE	ファイル名の関連付けが、不十分または不正である。
SE_ERR_DDEBUSY	他のDDE変換が処理されているため、DDE変換が処理できない。
SE_ERR_DDEFAIL	DDE変換が失敗した。
SE_ERR_DDETIMEOUT	DDE変換が要求時間切れで処理されなかった。
SE_ERR_DLLNOTFOUND	Windows 95のみ。指定されたダイナミックリンクライブラリが見つからない。
SE_ERR_FNF	Windows 95のみ。指定されたファイルが見つからない。
SE_ERR_NOASSOC	ファイル名の拡張子として関連付けられたアプリケーションがない。
SE_ERR_OOM	Windows 95のみ。処理するために十分なメモリがない。
SE_ERR_PNF	Windows 95のみ。指定されたパスが見つからない。
SE_ERR_SHARE	共有違反が発生した。

解　説

ShellExecute関数は、指定されたファイルをオープン、または表示する。ファイルには、実行可能ファイルか、文書ファイルが指定できる。ShellExecuteEx関数も参照すること。
*lpFile*パラメータで指定するファイルには、文書ファイルまたは実行可能ファイルを指定できる。文書ファイルの場合、ShellExecute関数は*lpOperation*パラメータの値に応じて、ファイルのオープンまたは表示を行う。実行可能ファイルの場合には、*lpOperation*パラメータに表示が指定されていても、ファイルは表示されずにオープンされる。

Windows 95の場合、ShellExecuteを使いWindows 95のフォルダをオープンやエクスポートすることができる。フォルダのオープンは、以下の様に行う。

```
ShellExecute(handle,NULL,path_to_folder,NULL,NULL,SW_SHOWNORMAL);
```

または

```
ShellExecute(handle,"open",path_to_folder,NULL,NULL,SW_SHOWNORMAL);
```

フォルダのエクスポートは、以下の様に行なう。

```
ShellExecute(handle,"explore",path_to_folder,NULL,NULL,SW_SHOWNORMAL);
```

*lpOperation*がNULLのとき、*lpFile*で指定されたファイルを対象とする。*lpOperation*が「オープン」または、「エクスポート」のときは、強制処理する。

参　照　→FindExecutable,ShellExecuteEx

2. 5 Graphic Device Interface
GDIインターフェイス

Windowsプログラミングインターフェイスでのもっとも大きな位置を示すGUIに関する処理のほとんどを司るのがグラフィックデバイスインターフェイス(GDI)関数群である。MS-DOSともっとも大きくことなり、もっともWindowsらしい関数群である。

Bitmap

BitBlt	gdi32.dll	95	NT
矩形カラーデータの移動			

V C

```
BOOL BitBlt(
    HDC      hdcDest,   //転送先のデバイスコンテキストのハンドル
    int      nXDest,    //転送先の矩形の左上隅のx座標
    int      nYDest,    //転送先の矩形の左上隅のy座標
    int      nWidth,    //転送先の矩形の幅
    int      nHeight,   //転送先の矩形の高さ
    HDC      hdcSrc,    //転送元のデバイスコンテキストのハンドル
    int      nXSrc,     //転送元の矩形の左上隅のx座標
    int      nYSrc,     //転送元の矩形の左上隅のy座標
    DWORD    dwRop      //ラスタオペレーションコード
    );
```

V B

```
Declare Function BitBlt Lib "gdi32" Alias "BitBlt" (ByVal hDestDC As Long, ByVal x As Long, ByVal y As Long, ByVal nWidth As Long, ByVal nHeight As Long, ByVal hSrcDC As Long, ByVal xSrc As Long, ByVal ySrc As Long, ByVal dwRop As Long) As Long
```

パラメータ

- *hdcDest*　転送先のデバイスコンテキストの識別子。
- *nXDest*　転送先の矩形の左上隅の論理x座標を指定する。
- *nYDest*　転送先の矩形の左上隅の論理y座標を指定する。
- *nWidth*　転送先と転送元の矩形の幅を論理単位で指定する。
- *nHeight*　転送先と転送元の矩形の高さを論理単位で指定する。

・*hdcSrc*　転送元のデバイスコンテキストの識別子。
・*nXSrc*　転送元の矩形の左上隅の論理x座標を指定する。
・*nYSrc*　転送元の矩形の左上隅の論理y座標を指定する。
・*dwRop*　ラスタオペレーションコードを指定する。ラスタオペレーションコードとは、転送元の矩形の色データを、転送先の矩形の色データに結合する方法を定義したものである。
共通のラスタオペレーションコードは以下の通り。

値	説明
BLACKNESS	物理パターンのインデックス0の色で転送先の矩形を塗りつぶす(この色は、デフォルト物理パターンでは黒である)。
DSTINVERT	転送先の矩形を反転する。
MERGECOPY	指定されたパターンビットマップと転送元のビットマップをビットごとのAND演算で結合する。
MERGEPAINT	反転した転送元のビットマップと転送先のビットマップをビットごとのOR演算で結合する。
NOTSRCCOPY	反転した転送元のビットマップを転送先にコピーする。
NOTSRCERASE	転送先のビットマップと転送元のビットマップをビットごとのOR演算で結合し、その結果を反転させる。
PATCOPY	指定されたパターンビットマップを転送先のビットマップにコピーする。
PATINVERT	転送先のビットマップと指定されたパターンビットマップをビットごとのXOR演算で結合する。
PATPAINT	反転した転送元のビットマップとパターンビットマップをビットごとのOR演算で結合し、この結果と転送先のビットマップをビットごとのOR演算で結合する。
SRCAND	転送先のビットマップと転送元のビットマップのピクセルをビットごとのAND演算で結合する。
SRCCOPY	転送元のビットマップを転送先のビットマップにコピーする。
SRCERASE	反転した転送先のビットマップと転送元のビットマップをビットごとのAND演算で結合する。
SRCINVERT	転送先のビットマップと転送元のビットマップのピクセルをビットごとのXOR演算で結合する。
SRCPAINT	転送先のビットマップと転送元のビットマップのピクセルをビットごとのOR演算で結合する。
WHITENESS	物理パターンのインデックス1の色で転送先の矩形を塗りつぶす(この色は、デフォルト物理パターンでは白である)。

戻り値

正常終了	TRUE
異常終了	FALSE

詳細なエラー情報は、GetLastError関数で取得。

解　説

BitBlt関数は、指定されたデバイスコンテキストから転送先のデバイスコンテキストに、ピクセル単位の矩形のカラーデータを移動する。

配置替えや分割の変換が、転送元のデバイスコンテキストに影響するとき、BitBlt関数はエラーを返す。他の変換が転送元のデバイスコンテキストに存在するとき(かつ、その変換が転送先のデバイスコンテキストに影響しないとき)、転送先のデバイスコンテキストの矩形は必要に応じて縮小や拡大する。
転送元と転送先のデバイスコンテキストのカラーフォーマットが異なる場合、BitBlt関数は転送元のカラーフォーマットを転送先のフォーマットに変換して合わせる。
エンハンストメタファイルが記録しているときに、転送元デバイスコンテキストがエンハンストメタファイルのデバイスコンテキストを認識するとエラーとなる。
すべてのデバイスでBitBlt関数がサポートされているわけではない。GetDeviceCaps関数でRC_BITBLTラスタを指定することにより、詳しい情報を調べること。
転送元と転送先のデバイスコンテキストが違うデバイスであるとき、BitBlt関数はエラーを返す。

参　照　→GetDeviceCaps,MaskBlt,PlgBlt,→StretchBlt

CreateBitmap	gdi32.dll	95	NT
ビットマップ作成			

V　C

```
HBITMAP CreateBitmap(
    int             nWidth,      //ビットマップの幅(ピクセル)
    int             nHeight,     //ビットマップの高さ(ピクセル)
    UINT            cPlanes,     //デバイスが使用するカラープレーンの数
    UINT            cBitsPerPel, //単一ピクセル当たりのカラービットの数
    CONST VOID *lpvBits          //カラーデータを持つ配列のポインタ
    );
```

V　B

Declare Function CreateBitmap Lib "gdi32" Alias "CreateBitmap" (ByVal nWidth As Long, ByVal nHeight As Long, ByVal nPlanes As Long, ByVal nBitCount As Long, lpBits As Any) As Long

パラメータ

- *nWidth*　ビットマップの幅をピクセル単位で指定する。
- *nHeight*　ビットマップの高さをピクセル単位で指定する。
- *cPlanes*　デバイスにより使われるビットマップ中のカラープレーンの数を指定する。
- *cBitsPerPel*　単一ピクセル当たりのカラービットの数を指定する。
- *lpvBits*　ピクセル単位の矩形の色を設定するために使う、カラーデータの配列へのポインタ。矩形の走査線ごとにWORD単位で並ぶ(WORD単位で並んでいない走査線は、0でうめられる)。このパラメータがNULLのときには、新しいビットマップは定義されない。

戻り値

正常終了	ビットマップハンドル
異常終了	NULL

解　説

CreateBitmap関数は、指定された幅、高さ、カラーフォーマット(カラー計画とピクセル当たりのビット数)を持つビットマップを作成する。

作成されたビットマップは、SelectObject関数を使用してデバイスコンテキストとして選択できる。

アプリケーションのパフォーマンス向上のために、CreateBitmap関数はカラービットマップを作成するために使用するのが望ましい。モノクロビットマップを作成する場合には、CreateCompatibleBitmap関数を使用する。

カラービットマップがデバイスコンテキストより選択されて、CreateBitmap関数より戻るとき、Windowsは選択されたデバイスコンテキストのフォーマットに適合したビットマップであることを保証する。

CreateCompatibleBitmapはデバイスコンテキストを取得し、指定されたデバイスコンテキストと同じフォーマットのビットマップを返す。この理由により、SelectObject関数を呼び出した後で、カラービットマップを返すのはCreateBitmap関数より早い。

ビットマップがモノクロのときは、0が前景色に、1が転送先のデバイスコンテキストの背景色となる。

アプリケーションが*nWidth*か*nHeight*パラメータに0を設定すると、CreateBitmap関数はモノクロビットマップのハンドルを返す。ビットマップが必要なくなったら、DeleteObject関数を使用してビットマップを削除する。

参　照

CreateBitmapIndirect,→CreateCompatibleBitmap,→CreateDIBitmap,DeleteObject,
→GetBitmapBits,SelectObject,→SetBitmapBits

CreateCompatibleBitmap	gdi32.dll	95	NT
デバイス互換ビットマップ作成			

V　C

```
HBITMAP CreateCompatibleBitmap(
    HDC     hdc,        //デバイスコンテキストのハンドル
    int     nWidth,     //ビットマップの幅(ピクセル)
    int     nHeight     //ビットマップの高さ(ピクセル)
    );
```

V　B

```
Declare Function CreateCompatibleBitmap Lib "gdi32" Alias "CreateCompatibleBitmap" (ByVal hdc As Long, ByVal nWidth As Long, ByVal nHeight As Long) As Long
```

パラメータ

- *hdc*　デバイスコンテキストの識別子。
- *nWidth*　ビットマップの幅をビット単位で指定する。
- *nHeight*　ビットマップの高さをビット単位で指定する。

戻り値

正常終了	ビットマップハンドル
異常終了	NULL

解　説

CreateCompatibleBitmap関数は、指定されたデバイスコンテキストに関連するデバイスと互換性のあるビットマップを作成する。
CreateCompatibleBitmap関数により作成されたカラーの形式は、*hdc*パラメータで識別されたデバイスと同じカラーの形式を持つ。このビットマップは、オリジナルデバイスと互換性のある任意のメモリデバイスコンテキストより選択できる。
メモリデバイスコンテキストでは、カラーのビットマップもモノクロのビットマップも選択できる。このため指定されたデバイスコンテキストがメモリデバイスコンテキストのとき、CreateCompatibleBitmap関数が返すビットマップの形式と異なる。しかし、メモリデバイスコンテキスト以外のデバイスコンテキストの場合には、そのデバイスと互換性のあるビットマップの形式は常に指定されたデバイスコンテキストカラーの形式やカラーパターンと同じになる。
アプリケーションが*nWidth*か*nHeight*パラメータに0を設定すると、CreateCompatibleBitmap関数は、モノクロビットマップのハンドルを返す。
*hdc*パラメータにより識別されるデバイスコンテキストから選択されたビットマップがCreateDIBSection関数で作成されたDIBセクションのとき、CreateCompatibleBitmap関数はDIBセクションを作成する。
ビットマップが必要なくなったとき、DeleteObject関数を使用してビットマップを削除する。

参　照

CreateDIBSection,DeleteObject,SelectObject

CreateDIBitmap	gdi32.dll	95	NT
デバイス固有メモリビットマップ作成			

V　C

```
HBITMAP CreateDIBitmap(
    HDC                        hdc,      //デバイスコンテキストのハンドル
    CONST BITMAPINFOHEADER*    lpbmih,   //ビットマップのサイズとフォーマ
                                           ットデータへのポインタ
    DWORD                      fdwInit,  //初期化フラグ
    CONST VOID*                lpbInit,  //初期化データへのポインタ
    CONST BITMAPINFO*          lpbmi,    //ビットマップのカラー形式データ
                                           へのポインタ
    UINT                       fuUsage   //カラーデータ
    );
```

V　B

```
Declare Function CreateDIBitmap Lib "gdi32" Alias "CreateDIBitmap" (ByVal hdc As Long, lpInfoHeader As BITMAPINFOHEADER, ByVal dwUsage As Long, lpInitBits As Any, lpInitInfo As BITMAPINFO, ByVal wUsage As Long) As Long
```

パラメータ

・*hdc*　　デバイスコンテキストの識別子。

・*lpbmih*　　BITMAPINFOHEADER構造体へのポインタ。このパラメータは、CreateDIBitmap関数による。

・*fdwInit*　　ビットマップビットをオペレーションシステムが初期化する方法を指定するビットフラグの設定。以下の値がビットフラグに指定できる。

値	説明
CBM_INIT	オペレーティングシステムが*lpbInit*パラメータと*lpbmi*パラメータに指定されたビットを使用してビットマップを初期化することを設定する。*fdwInit*が0のとき、オペレーティングシステムは、ビットマップのビットを初期化しない。

・*lpbInit*　　ビットマップデータの初期値を格納したバイト配列へのポインタ。ビットマップ値の形式は、*lpbmi*パラメータに指定されたBITMAP INFOHEADER構造体の*biBitCount*メンバにより決まる。

・*lpbmi*　　*lpbInit*パラメータにより示される配列の寸法とカラー形式を記述する、BITMAPINFO構造体へのポインタ。

・*fuUsage*　　BITMAPINFO構造体の*bmiColors*メンバが、赤、緑、青の(RGB)値かパレットのインデックスのどちらを表すかを指定する。*fuUsage*パラメータには、以下の値のいずれかを指定する。

値	説明
DIB_PAL_COLORS	カラーテーブルは、選択されたビットマップのデバイスコンテキストの論理パレットへの16ビットインデックスの配列である。
DIB_RGB_COLORS	カラーテーブルは、RGB値である。

戻り値

正常終了	ビットマップハンドル
異常終了	NULL

解　説

CreateDIBitmap関数は、デバイスに依存しないビットマップ(DIB)からデバイス固有のメモリビットマップを作成する。必要であれば、ビットマップ内のビットを設定する。

*fdwInit*パラメータがCBM_CREATDIBフラグのとき、これ以上サポートされない。

ビットマップが必要なくなったとき、DeleteObject関数で削除する。

参　照

→BITMAPINFOHEADER,→BITMAPINFO,DeleteObject,→GetDeviceCaps,
→GetSystemPaletteEntries,SelectObject

ExtFloodFill	gdi32.dll	95	NT
指定範囲塗りつぶし			

V C

```
BOOL ExtFloodFill(
    HDC        hdc,          //デバイスコンテキストのハンドル
    int        nXStart,      //塗りつぶしの開始x座標
    int        nYStart,      //塗りつぶしの開始y座標
    COLORREF   crColor,      //塗りつぶす色
    UINT       fuFillType    //塗りつぶしのタイプ
    );
```

V B

Declare Function ExtFloodFill Lib "gdi32" Alias "ExtFloodFill" (ByVal hdc As Long, ByVal x As Long, ByVal y As Long, ByVal crColor As Long, ByVal wFillType As Long) As Long

パラメータ

- *hdc* デバイスコンテキストの識別子。
- *nXStart* 塗りつぶしを開始する点の論理x座標を指定する。
- *nYStart* 塗りつぶしを開始する点の論理y座標を指定する。
- *crColor* 塗りつぶす領域または境界の色を指定する。*crColor*の意味は*fuFillType*パラメータの値により決まる。
- *fuFillType* 塗りつぶし操作のタイプを指定する。以下の値のいずれかになる。

値	説明
FLOODFILLBORDER	塗りつぶす領域が、*crColor*パラメータで指定された色により囲まれている必要がある。このスタイルは、FloodFill関数による塗りつぶしと同じである。
FLOODFILLSURFACE	*crColor*で指定された色の領域を塗りつぶす。指定された色が存在する限り、すべての方向に塗りつぶしが続けられる。複数の色からなる境界の場合には、このスタイルが便利である。

戻り値

正常終了	TRUE
異常終了	FALSE

詳細なエラー情報は、GetLastErrorにより取得。

解説

ExtFloodFill関数は、画面上の指定された範囲をカレントブラシで塗りつぶす。

以下の場合には、この関数は失敗する。

- 塗りつぶしが完了しなかった。
- 指定された点の色が*crColor*パラメータで指定された境界色である(FLOODFILLBORDERを指定したとき)。
- 指定された点の色が*crColor*パラメータで指定された色でない(FLOODFILLSURFACEを指定したとき)。
- クリッピングリージョンの外の点が指定された。

*fuFillType*パラメータがFLOODFILLBORDERの場合、Windowsは塗りつぶす領域が*crColor*パラメータで指定される色で完全に囲まれているものとみなす。この関数は、*nXStart*パラメータと*nYStart*パラメータにより指定される座標からすべての方向に塗りつぶしを始め、指定された色の付いた境界まで塗りつぶしを続ける。

*fuFillType*パラメータがFLOODFILLSURFACEの場合には、Windowsは単色で領域を塗りつぶすとみなす。この関数は*nXStart*および*nYStart*により指定される座標からすべての方向に塗りつぶしを始め、*crColor*で指定された色を含む隣接した領域を塗りつぶし続ける。

メモリデバイスコンテキスト、およびラスタディスプレイをサポートするデバイスだけがExtFloodFill関数をサポートする。サポートするデバイスの説明は、GetDeviceCaps関数を参照。

参照　FloodFill,GetDeviceCaps

FloodFill	gdi32.dll	95	NT
スクリーン領域塗りつぶし			

VC

```
BOOL FloodFill(
    HDC         hdc,        //デバイスコンテキストのハンドル
    int         nXStart,    //始点のx座標
    int         nYStart,    //始点のy座標
    COLORREF crFill        //塗りつぶしの色
  );
```

VB

```
Declare Function FloodFill Lib "gdi32" Alias "FloodFill" (ByVal hdc As Long, ByVal x As Long, ByVal y As Long, ByVal crColor As Long) As Long
```

パラメータ

- *hdc*　デバイスコンテキストの識別子。
- *nXStart*　塗りつぶしを始める点の論理x座標を指定する。
- *nYStart*　塗りつぶしを始める点の論理y座標を指定する。
- *crFill*　塗りつぶし領域の境界の色を指定する。

戻り値

正常終了	TRUE
異常終了	FALSE

詳細なエラー情報は、GetLastError関数により取得。

解　説

FloodFill関数は、カレントブラシを使用して、スクリーン表面の領域を塗りつぶす。塗りつぶされる領域は、*crFill*パラメータの指定通りに囲まれているものとみなされる。

FloodFill関数はWindowsの初期バージョンとの互換性のためにのみ存在する。Win32ベースのアプリケーションは、FLOODFILLBORDERを指定してExtFloodFill関数を使用すること。

以下の場合には、この関数は失敗する。

- 塗りつぶしが完了しなかった。
- 指定された点の色が*crFill*パラメータで指定された境界色である。
- クリッピングリージョンの外の点が指定された。

参　照

→ExtFloodFill

GetBitmapBits	gdi32.dll	95	NT
ビットマップビットのコピー			

V C

```
LONG GetBitmapBits(
    HBITMAP   hbmp,       //ビットマップのハンドル
    LONG      cbBuffer,   //バッファにコピーされるバイト数
    LPVOID    lpvBits     //ビットマップを受け取るバッファへのポインタ
    );
```

V B

```
Declare Function GetBitmapBits Lib "gdi32" Alias "GetBitmapBits" (ByVal hBitmap As Long, ByVal dwCount As Long, lpBits As Any) As Long
```

パラメータ

- *hbmp*　ビットマップの識別子。
- *cbBuffer*　コピーするバイト数を指定する。
- *lpvBits*　ビットマップのビットを受け取るバッファへのポインタ。ビットマップはバイトの配列である。

戻り値

正常終了	ビットマップサイズ（バイト単位）
異常終了	0

詳細なエラー情報は、GetLastError関数により取得。

解　説

GetBitmapBits関数は、指定されたビットマップをバッファにコピーする。

GetBitmapBits関数は、Win32APIの関数ではない。この関数は、Windowsの16ビットバージョンの関数である。Win32ベースでは、アプリケーションはGetDIBits関数を使用すること。

バッファの中のビットは、WORD並びである。言い換えれば、バッファはメモリ上の偶数アドレスから始まる。

参　照

→GetDIBits

GetBitmapDimensionEx	gdi32.dll	95	NT
ビットマップサイズ取得			

V C

```
BOOL GetBitmapDimensionEx(
    HBITMAP    hBitmap,        //ビットマップのハンドル
    LPSIZE     lpDimension     //大きさの構造体へのポインタ
    );
```

V B

Declare Function GetBitmapDimensionEx Lib "gdi32" Alias "GetBitmapDimensionEx" (ByVal hBitmap As Long, lpDimension As SIZE) As Long

パラメータ

- *hBitmap*　ビットマップの識別子。
- *lpDimension*　寸法が返されるSIZE構造体へのポインタ。

戻り値

正常終了	TRUE
異常終了	FALSE

詳細なエラー情報は、GetLastError関数により取得。

解　説

GetBitmapDimensionEx関数は、ビットマップの寸法を返す。この寸法は、SetBitmapDimensionEx関数により設定される。GetBitmapDimensionEx関数は、GetBitmapDimension関数の代わりとなる。
この関数は、ビットマップの高さと幅の入った構造体を返す。寸法が設定されていないときは、構造体のそれらの値に0を設定して返す。

参　照

SetBitmapDimensionEx,SIZE

GetDIBColorTable	gdi32.dll	95	NT
RGBカラー値の回復			

V　C

```
UINT GetDIBColorTable(
    HDC      hdc,         //デバイスコンテキストのハンドル
    UINT     uStartIndex, //回復するための最初の入力のカラーテーブルイ
                            ンデックス
    UINT     cEntries,    //回復するための最初の入力のカラーテーブルの数
    RGBQUAD *pColors      //カラーテーブルを受け取るバッファへのポインタ
);
```

V　B

```
Declare Function GetDIBColorTable Lib "gdi32" Alias "GetDIBColorTable" (ByVal hDC As Long, ByVal un1 As Long, ByVal un2 As Long, pRGBQuad As RGBQUAD) As Long
```

パラメータ

・*hdc*　デバイスコンテキストを指定する。DIBセクションのビットマップは、このデバイスコンテキストより選択されなければならない。

・*uStartIndex*　回復する最初のカラーテーブル入力を指定する。0から始まるカラーテーブルインデックス。

・*cEntries*　回復するカラーテーブル入力の数を指定する。

・*pColors*　DIBのカラーテーブルからのカラー情報を入れるRGBQUAD構造体の配列を受け取るバッファへのポインタ。このバッファは、RGBQUAD構造体を*cEntries*の値だけ入れるのに十分な大きさが必要である。

戻り値

正常終了	カラーテーブル入力数
異常終了	0

詳細なエラー情報はGetLastError関数により取得。

解　説

GetDIBColorTable関数は、指定されたデバイスコンテキストで選択されたDIBセクションビットマップのカラーテーブルの入力範囲からRGB(赤、緑、青)のカラー値を回復する。

GetDIBColorTable関数は、ピクセル当たり1、4または8ビットを使うDIBセクションビットマップのカラーテーブルを回復するために呼び出される。BITMAPINFOHEADER構造体のbiBitCountメンバが、ピクセル当たりのビット数を示す。biBitCountの値が8より大きいDIBセクションビットマップは、カラーテーブルを持たないがカラーマスクに関連する。カラーマスクの回復は、GetObject関数を呼び出す。

参　照

→BITMAPINFOHEADER,CreateDIBSection,→DIBSECTION,GetObject,
→RGBQUAD,→SetDIBColorTable

GetDIBits	gdi32.dll	95	NT
ビットマップの取得コピー			

V C

```
int GetDIBits(
    HDC           hdc,         //デバイスコンテキストのハンドル
    HBITMAP       hbmp,        //ビットマップのハンドル
    UINT          uStartScan,  //転送先ビットマップ内に設定される最初
                               の走査線
    UINT          cScanLines,  //コピーする走査線の数
    LPVOID        lpvBits,     //ビットマップビットの配列のアドレス
    LPBITMAPINFO  lpbi,        //ビットマップデータの構造体のアドレス
    UINT          uUsage       //RGBまたは、パターンのインデックス
    );
```

V B

```
Declare Function GetDIBits Lib "gdi32" Alias "GetDIBits" (ByVal aHDC As Long, ByVal hBitmap As Long, ByVal nStartScan As Long, ByVal nNumScans As Long, lpBits As Any, lpBI As BITMAPINFO, ByVal wUsage As Long) As Long
```

パラメータ

- *hdc*　デバイスコンテキストの識別子。
- *hbmp*　ビットマップの識別子。
- *uStartScan*　設定される最初の走査線を指定する。
- *cScanLines*　コピーする走査線の数を指定する。
- *lpvBits*　ビットマップデータを受け取るバッファのポインタ。このパラメータがNULLのとき、この関数は*lpbi*パラメータが指すBITMAPINFO構造体にビットマップの大きさやフォーマットを設定する。
- *lpbi*　デバイスに依存しないビットマップ(DIB)データのフォーマットを指定するBITMAPINFO構造体へのポインタ。
- *uUsage*　BITMAPINFO構造体のbmiColorsメンバのフォーマットを指定する。このパラメータは、以下の値のいずれかである。

値	意味
DIB_PAL_COLORS	カラーテーブルは、カレント論理パレットへの16ビットのインデックスの配列である。
DIB_RGB_COLORS	カラーテーブルは、赤、緑、青(RGB)の値である。

戻り値

正常終了(Windows 95) *lpvBits*=0	ビットマップの走査線数
正常終了(WindowsNT) *lpvBits*=0	0以外
異常終了	0

解　説

GetDIBits関数は、指定されたビットマップのビットを取得し、それをデバイスに依存しない形式で*lpvBits*パラメータが指すバッファにコピーする。
DIBのために要求された形式が内部形式に合っていれば、ビットマップのRGB値がコピーされる。要求された形式が内部形式に合っていないと、カラーテーブルが統合される。以下がその値である。

値	意味
1_BPP	カラーテーブルは、黒と白の入力からなる。
4_BPP	カラーテーブルは、標準VGAパターンと同じ色を混ぜたものからなる。
8_BPP	カラーテーブルは、GDIで定義された256色を混ぜたものからなる(256色には、デフォルト論理パレットの20色が含まれる)。
24_BPP	カラーテーブルは返されない。

*lpvBits*パラメータがポインタが設定されているとき、BITMAPINFOHEADER構造体の最初の6メンバは、DIBのサイズと形式を指定して初期化する。ボトムアップDIBは正の数で高さを設定し、トップダウンDIBは負の数で高さを設定する。このビットマップのカラーテーブルは、BITMAPINFO構造体に追加される。
*lpvBits*パラメータがNULLのとき、GetDIBits関数は*lpbi*で示される最初の構造体の最初のメンバを検査する。このメンバには、BITMAPINFOHEADERまたはBITMAPCOREHEADER構造体のバイト単位のサイズを指定する。この関数は、残りのメンバの初期化の方法を決定するため指定されたサイズを使う。
*lpvBits*パラメータがNULLで、BITMAPINFO構造体の*biBitCount*メンバが0で初期化されていてかつ、カラーテーブルがないとき、GetDIBits関数はBITMAPINFOHEADERまたはBITMAPCOREHEADERで塗りつぶす。このテクニックは、ビットマップの属性を確認するために使う。
アプリケーションがこの関数を呼び出すとき、*hbmp*パラメータにより認識されるビットマップは、デバイスコンテキストの中を選択してはならない。ボトムアップDIBの原点はビットマップの左下隅である。トップダウンDIBの原点は左上隅である。

参　照

→BITMAPCOREHEADER,→BITMAPINFO,→BITMAPINFOHEADER,→SetDIBits

GetPixel	gdi32.dll	95	NT
指定位置のカラー値取得			

V C

```
COLORREF GetPixel(
        HDC     hdc,      //デバイスコンテキストのハンドル
        int     nXPos,    //ピクセルのx座標
        int     nYPos     //ピクセルのy座標
        );
```

V B

```
Declare Function GetPixel Lib "gdi32" Alias "GetPixel" (ByVal hdc As Long, ByVal x As Long, ByVal y As Long) As Long
```

パラメータ

- *hdc*　　デバイスコンテキストの識別子。
- *nXPos*　　調査するピクセルの論理x座標を指定する。
- *nYPos*　　調査するピクセルの論理y座標を指定する。

戻り値

正常終了	RGB値
ピクセルがクリッピング領域外	CLR_INVALID

解　説

GetPixel関数は、指定された位置にあるピクセルの赤、緑、青(RGB)カラー値を取得する。

ピクセルは、カレントクリッピング領域内になければならない。

GetPixel関数をサポートしていないデバイスもある。アプリケーションは、指定されたデバイスがこの関数をサポートしているデバイスかGetDeviceCaps関数を呼び出し確認すること。

参　照

→GetDeviceCaps

GetStretchBltMode	gdi32.dll	95	NT
ビットマップ伸縮モード取得			

V C

```
int GetStretchBltMode(
    HDC     hdc     //デバイスコンテキストのハンドル
    );
```

V B

Declare Function GetStretchBltMode Lib "gdi32" Alias "GetStretchBltMode" (ByVal hdc As Long) As Long

パラメータ

・*hdc*　デバイスコンテキストの識別子。

戻り値

正常終了	ビットマップ伸縮モード
異常終了	0

解　説

GetStretchBltMode関数は、カレントビットマップ伸縮モードを取得する。ビットマップ伸縮モードとは、StretchBlt関数を使用してビットマップを圧縮するときに、情報を削除する方法を定義するものである。

参　照

→SetStretchBltMode,→StretchBlt

LoadBitmap	user32.dll	95	NT
ビットマップリソースロード			

V C

```
HBITMAP LoadBitmap(
      HINSTANCE  hInstance,      //アプリケーションインスタンスのハンドル
      LPCTSTR    lpBitmapName    //ビットマップリソース名へのポインタ
      );
```

V B

Declare Function LoadBitmap Lib "user32" Alias "LoadBitmapA" (ByVal hInstance As Long, ByVal lpBitmapName As String) As Long

パラメータ

・*hInstance*　ロードするビットマップを持つ実行可能ファイルを含むモジュールのインスタンスの識別子。

・*lpBitmapName*　ロードするビットマップリソースの名前のNULLで終わる文字列へのポインタ。また、このパラメータには、下位ワードにリソース識別子を設定し、上位ワードに0を設定することもできる。この値は、MAKEINTRESOURCEマクロを使用して作成できる。

戻り値

正常終了	ビットマップのハンドル
異常終了	NULL

解　説

LoadBitmap関数は、モジュールの実行可能ファイルから、指定されたビットマップリソースをロードする。
*lpBitmapName*が示すビットマップがない場合や、ビットマップをロードするためのメモリが不足している場合、関数は失敗する。
アプリケーションはLoadBitmap関数を使用して、Win32APIが使用する定義済みビットマップにアクセスできる。定義済みビットマップにアクセスするには、アプリケーションは*hInstance*パラメータにNULLを設定し、*lpBitmapName*パラメータに以下の値のいずれかを設定する。

OBM_BTNCORNERS	OBM_MNARROW	OBM_RESTORED
OBM_BTSIZE	OBM_OLD_CLOSE	OBM_RGARROW
OBM_CHECK	OBM_OLD_DNARROW	OBM_RGARROWD
OBM_CHECKBOXES	OBM_OLD_LFARROW	OBM_RGARROWI
OBM_CLOSE	OBM_OLD_REDUCE	OBM_SIZE
OBM_COMBO	OBM_OLD_RESTORE	OBM_UPARROW
OBM_DNARROW	OBM_OLD_RGARROW	OBM_UPARROWD
OBM_DNARROWD	OBM_OLD_UPARROW	OBM_UPARROWI
OBM_DNARROWI	OBM_OLD_ZOOM	OBM_ZOOM
OBM_LFARROW	OBM_REDUCE	OBM_ZOOMD
OBM_LFARROWD	OBM_REDUCED	
OBM_LFARROWI	OBM_RESTORE	

OBM_OLDで始まるビットマップ名は、Windows 3.0以前のバージョンで使われていたビットマップを表している。
OBM_定数を使うアプリケーションでは、WINDOWS.Hヘッダーファイルをインクルードする前にOEMRESOURCE定数を定義する。アプリケーションは、DeleteObject関数を呼び出して、LoadBitmap関数が返した各ビットマップハンドルを削除しなければならない。

参　照

→CreateBitmap,DeleteObject,LoadCursor,LoadIcon,→MAKEINTRESOURCE

SetBitmapBits	gdi32.dll	95	NT
ビットマップから色データ設定			

V C

```
LONG SetBitmapBits(
    HBITMAP      hbmp,    //ビットマップのハンドル
    DWORD        cBytes,  //ビットマップ配列内のバイト数
    CONST VOID* lpBits    //ビットマップビットを格納している配列へのポインタ
    );
```

V B

```
Declare Function SetBitmapBits Lib "gdi32" Alias "SetBitmapBits" (ByVal hBitmap As Long, ByVal dwCount As Long, lpBits As Any) As Long
```

パラメータ

・*hbmp*　　設定するビットマップの識別子。
・*cBytes*　　*lpBits*パラメータが指すビットのバイト数を指定する。
・*lpBits*　　指定されるビットマップの色データのバイト配列へのポインタ。

戻り値

正常終了	設定で利用したサイズ（バイト単位）
異常終了	0

解　説

SetBitmapBits関数は、指定されたビットマップのビットを色データのビット値に設定する。
SetBitmapBits関数は、Windowsの初期のバージョンに適合するために含まれている。Win32ベースのアプリケーションでは、SetDIBits関数を使用すること。
*lpBits*で示される配列は、WORD並びである。

参　照

→GetBitmapBits,→SetDIBits

SetBitmapDimensionEx	gdi32.dll	95	NT
ビットマップ優先サイズ割り当て			

V C

```
BOOL SetBitmapDimensionEx(
    HBITMAP   hBitmap,   //ビットマップのハンドル
    int       nWidth,    //0.1mm単位のビットマップの幅
    int       nHeight,   //0.1mm単位のビットマップの高さ
    LPSIZE    lpSize     //オリジナルの寸法が格納される構造体のアドレス
    );
```

V B

```
Declare Function SetBitmapDimensionEx Lib "gdi32" Alias "SetBitmapDimensionEx" (ByVal hbm As Long, ByVal nX As Long, ByVal nY As Long, lpSize As SIZE) As Long
```

パラメータ

- *hBitmap*　ビットマップの識別子。このビットマップは、DIBビットマップではない。
- *nWidth*　ビットマップの幅を0.1ミリメートル単位で指定する。
- *nHeight*　ビットマップの高さを0.1ミリメートル単位で指定する。
- *lpSize*　ビットマップの以前の寸法を格納するSIZE構造体へのポインタ。このポインタは、NULLでもかまわない。

戻り値

正常終了	TRUE
異常終了	FALSE

詳細なエラー情報は、GetLastError関数で取得。

解　説

SetBitmapDimensionEx関数は、ビットマップの優先される寸法を割り当てる。これらの寸法は、アプリケーションでのみ使われ、Windowsは使わない。
アプリケーションは、GetBitmapDimensionEx関数を呼び出すことによりSetBitmapDimensionEx関数でビットマップの割り当てられた寸法を回復することができる。
*hBitmap*で示されるビットマップは、CreateDIBSection関数で作られたビットマップであるDIBセクションではない。ビットマップがDIBセクションのとき、SetBitmapDimensionEx関数は失敗する。

参　照

CreateDIBSection,→GetBitmapDimensionEx,→SIZE

SetDIBColorTable	gdi32.dll	95	NT
ビットマップからカラー値設定			

V C

```
UINT SetDIBColorTable(
    HDC              hdc,          //デバイスコンテキストのハンドル
    UINT             uStartIndex,  //設定するための最初の入力のカラーテー
                                     ブルインデックス
    UINT             cEntries,     //設定するための最初の入力のカラーテー
                                     ブルの数
    CONST RGBQUAD*   pColors       //カラーテーブルの配列のポインタ
    );
```

V B

Declare Function SetDIBColorTable Lib "gdi32" Alias "SetDIBColorTable" (ByVal hDC As Long, ByVal un1 As Long, ByVal un2 As Long, pcRGBQuad As RGBQUAD) As Long

パラメータ

- *hdc*　デバイスコンテキストを指定する。デバイスに依存しないビットマップは、このデバイスコンテキストより選択されなければならない。
- *uStartIndex*　設定する最初のカラーテーブル入力を指定する、0から始まるカラーテーブルインデックス。
- *cEntries*　設定するカラーテーブル入力の数を指定する。
- *pColors*　DIBのカラーテーブルからの新しいカラー情報を入れるRGBQUAD構造体の配列へのポインタ。

戻り値

正常終了	カラーテーブル入力数
異常終了	0

詳細なエラー情報はGetLastError関数により取得。

解　説

SetDIBColorTable関数は、指定されたデバイスコンテキストで選択されたデバイスに依存しないビットマップ(DIB)のカラーテーブルの入力範囲から、RGB(赤、緑、青)のカラー値を設定する。

この関数は、ピクセル当たり1、4または8ビットを使うデバイスに依存しないビットマップのカラーテーブルを設定するために呼び出される。BITMAPINFOHEADER構造体のbiBitCountメンバが、ピクセル当たりのビット数を示す。*biBitCount*の値が8より大きいデバイスに依存しないビットマップは、カラーテーブルを持たない。

参　照

→BITMAPINFOHEADER,CreateDIBSection,→GetDIBColorTable,GetObject,
→DIBSECTION,→RGBQUAD

SetDIBits	gdi32.dll	95	NT
ビットマップからカラーデータ設定			

VC

```
int SetDIBits(
    HDC                   hdc,         //デバイスコンテキストのハンドル
    HBITMAP               hbmp,        //ビットマップのハンドル
    UINT                  uStartScan,  //開始走査線
    UINT                  cScanLines,  //開始走査線
    CONST VOID*           lpvBits,     //ビットマップビットの配列
    CONST BITMAPINFO*     lpbmi,       //ビットマップデータの構造体のアドレス
    UINT                  fuColorUse   //使用するカラーインデックスのタイプ
    );
```

VB

```
Declare Function SetDIBits Lib "gdi32" Alias "SetDIBits" (ByVal hdc As Long, ByVal hBitmap As Long, ByVal nStartScan As Long, ByVal nNumScans As Long, lpBits As Any, lpBI As BITMAPINFO, ByVal wUsage As Long) As Long
```

パラメータ

- *hdc*　デバイスコンテキストの識別子。
- *hbmp*　指定されたDIBよりカラーデータを使用するビットマップの識別子。
- *uStartScan*　*lpvBits*パラメータが指す配列のデバイスに依存しないカラーデータの最初の走査線を指定する。
- *cScanLines*　デバイスに依存しないカラーデータを含む配列の走査線数を指定する。
- *lpvBits*　バイト配列として格納されている、DIBカラーデータへのポインタ。ビットマップ値の形式は、*lpbmi*パラメータが指すBITMAPINFO構造体の最初のメンバである*biBitCount*メンバの値により異なる。
- *lpbmi*　DIBについての情報を持つ、BITMAPINFO構造体へのポインタ。
- *fuColorUse*　BITMAPINFO構造体のbmiColorsメンバに、赤、緑、青(RGB)の値が格納されているのか、またはパレットへのインデックスが格納されているのかを指定する。このパラメータは、以下の値のいずれかになる。

値	意味
DIB_PAL_COLORS	カラーテーブルは、*hdc*パラメータが識別するデバイスコンテキストの論理パレットへの、16ビットインデックスの配列である。
DIB_RGB_COLORS	カラーテーブルはRGB値である。

戻り値

正常終了	コピーされた走査線数
異常終了	0

詳細なエラー情報は、GetLastError関数により取得。

解　説

SetDIBits関数は、ビットマップのピクセルをデバイスに依存しないビットマップ(DIB)で指定されるカラーデータに設定する。
ビットマップを描画するときの速度は、各ビットマップビットがシステムパレットを参照しているときに最適化される。
アプリケーションはGetSystemPaletteEntries関数によりシステムパレットの色とインデックスを回復できる。色とインデックスの回復後、アプリケーションはDIBを作成できる。システムパレットについて詳しくはSystemPaletteを参照すること。
*fuColorUse*パラメータにDIB_PAL_COLORSが設定されているとき、デバイスコンテキストは*hdc*パラメータにより識別する。それ以外は無視される。
アプリケーションがこの関数を呼び出すとき、*hbmp*パラメータにより識別されるビットマップは、デバイスコンテキストの中を選択してはならない。
ボトムアップDIBの原点はビットマップの左下隅である。トップダウンDIBの原点は左上隅である。

参　照

→BITMAPINFO,→GetSystemPaletteEntries

SetDIBitsToDevice	gdi32.dll	95	NT
ビットマップから矩形ピクセルを設定			

V　C

```
int SetDIBitsToDevice(
    HDC                 hdc,         //デバイスコンテキストのハンドル
    int                 XDest,       //転送先の矩形の左上点のx座標
    int                 YDest,       //転送先の矩形の左上点のy座標
    DWORD               dwWidth,     //転送元の矩形の幅
    DWORD               dwHeight,    //転送元の矩形の高さ
    int                 XSrc,        //転送元の矩形の左下点のx座標
    int                 YSrc,        //転送元の矩形の左下点のy座標
    UINT                uStartScan,  //配列中の最初の走査線
    UINT                cScanLines,  //走査線の数
    CONST VOID*         lpvBits,     //DIBビットの配列のアドレス
    CONST BITMAPINFO*   lpbmi,       //ビットマップ情報の構造体へのポインタ
    UINT                fuColorUse   //RGB値またはパレットインデックス
    );
```

V　B

Declare Function SetDIBitsToDevice Lib "gdi32" Alias "SetDIBitsToDevice" (ByVal hdc As Long, ByVal x As Long, ByVal y As Long, ByVal dx As Long, ByVal dy As Long, ByVal SrcX As Long, ByVal SrcY As Long, ByVal Scan As Long, ByVal NumScans As Long, Bits As Any BitInfo As BITMAPINFO, By Val wUsage As Long) As Long

パラメータ

- *hdc*　デバイスコンテキストの識別子。
- *XDest*　転送先の矩形の左上点の論理単位のx座標を指定する。
- *YDest*　転送先の矩形の左上点の論理単位のy座標を指定する。
- *dwWidth*　DIBの幅を論理単位で指定する。
- *dwHeight*　DIBの高さを論理単位で指定する。
- *XSrc*　DIBの左下点の論理単位のx座標を指定する。
- *YSrc*　DIBの左下点の論理単位のy座標を指定する。
- *uStartScan*　DIBの開始走査線を指定する。
- *cScanLines*　*lpvBits*パラメータの配列内のDIB走査線数を指定する。
- *lpvBits*　バイト配列として格納されている、DIBカラーデータへのポインタ。
- *lpbmi*　DIBに関する情報を持つ、BITMAPINFO構造体へのポインタ。
- *fuColorUse*　BITMAPINFO構造体のbmiColorsメンバに、赤、緑、青(RGB)の値が格納されているのか、またはパレットへのインデックスが格納されているのかを指定する。このパラメータは、以下の値のいずれかになる。

値	意味
DIB_PAL_COLORS	カラーテーブルは、カレント論理パレットへの、16ビットインデックスの配列である。
DIB_RGB_COLORS	カラーテーブルはRGB値である。

戻り値

正常終了	設定された走査線数
異常終了	0

詳細なエラー情報は、GetLastError関数で取得。

解　説

SetDIBitsToDevice関数は、デバイスに依存しないビットマップ(DIB)から、カラーデータを使用するデバイスコンテキストに関連したデバイス上に指定された矩形にピクセルを設定する。
ビットマップを描画するときの速度は、各ビットマップビットがシステムパレット参照しているときに最適化される。
アプリケーションは、GetSystemPaletteEntries関数によりシステムパレットの色とインデックスを回復できる。色とインデックスの回復後、アプリケーションはDIBを作成できる。システムパレットについて詳しくはColorsを参照すること。
ボトムアップDIBの原点はビットマップの左下隅である。トップダウンDIBの原点は左上隅である。アプリケーションでデバイスの画面上にある大きなデバイスに依存しないビットマップからビットを設定するときは、SetDIBitsToDevice関数を繰り返し呼び出し、ビットマップ全体を別々に*lpvBits*バッファの中に読み取ることにより必要なメモリ量を節約できる。*uStartScan*パラメータと*cScanLines*パラメータは、*lpvBits*配列内にある、ビットマップ全体の特定の部分を識別する値になる。フォアグラウンドで全画面のMS-DOSセッションの実行中、バックグラウンドで実行している処理から呼ばれた場合、SetDIBitsToDevice関数はエラーを返す。

参　照

→BITMAPINFO,→GetSystemPaletteEntries,→SetDIBits,→StretchDIBits

SetPixel	gdi32.dll	95	NT
ピクセル色設定			

V　C

```
COLORREF SetPixel(
    HDC         hdc,      //デバイスコンテキストのハンドル
    int         X,        //ピクセルのx座標
    int         Y,        //ピクセルのy座標
    COLORREF    crColor   //ピクセルの色
    );
```

V　B

```
Declare Function SetPixel Lib "gdi32" Alias "SetPixel" (ByVal hdc As Long, ByVal x As Long, ByVal y As Long, ByVal crColor As Long) As Long
```

パラメータ

- *hdc*　デバイスコンテキストの識別子。
- *X*　設定する位置の論理x座標を指定する。
- *Y*　設定する位置の論理y座標を指定する。
- *crColor*　指定された位置に描画するときの色を指定する。

GDIインターフェイス

戻り値

正常終了	設定されたRGB値
異常終了	-1

詳細なエラー情報は、GetLastError関数により取得。

解　説

SetPixel関数は、指定された座標にあるピクセルを指定の色に設定する。
指定された点がクリッピングリージョン内にないときは、この関数は失敗する。
すべてのデバイスがSetPixel関数をサポートするとは限らない。GetDeviceCaps関数によりデバイスを調べること。

参　照

→GetDeviceCaps,→SetPixelV

SetPixelV

──────	95	NT

ピクセル色設定

V C

```
BOOL SetPixelV(
        HDC           hdc,        //デバイスコンテキストのハンドル
        int           X,          //ピクセルのx座標
        int           Y,          //ピクセルのy座標
        COLORREF      crColor     //新しいピクセルの色
        );
```

パラメータ

- *hdc*　　描画するために使う色を指定する。
- *X*　　設定する位置の論理x座標を指定する。
- *Y*　　設定する位置の論理y座標を指定する。
- *crColor*　　指定された位置に描画するときの色を指定する。

戻り値

正常終了	TRUE
異常終了	FALSE

詳細なエラー情報は、GetLastError関数で取得。

解　説

SetPixelV関数は、指定された座標に、指定された色にできるだけ近い色でピクセルを設定する。そのピクセルは、クリッピングリージョンとデバイスの表面の見える部分になければならない。すべてのデバイスがSetPixelV関数をサポートするとは限らない。GetDeviceCaps関数のRC_BITBLTで調査すること。
描画された点のカラー値を返す必要がないため、SetPixelV関数はSetPixel関数より早い。

参　照

→GetDeviceCaps,→SetPixel

SetStretchBltMode	gdi32.dll	95	NT
ビットマップ伸縮モード設定			

V C

```
int SetStretchBltMode(
    HDC    hdc,            //デバイスコンテキストのハンドル
    int    iStretchMode    //ビットマップの伸縮モード
    );
```

V B

Declare Function SetStretchBltMode Lib "gdi32" Alias "SetStretchBltMode" (ByVal hdc As Long, ByVal nStretchMode As Long) As Long

パラメータ

・*hdc* デバイスコンテキストの識別子。

・*iStretchMode* 伸縮モードを指定する。このパラメータは、以下の値のいずれかとなる。

値	意味
BLACKONWHITE	縮小化によってビットマップのラインが消去されるときに。AND演算子を用いてライン結合を行う。白黒のビットマップの場合、このモードでは、白色のピクセルが消えて黒色のピクセルが残る。
COLORONCOLOR	ピクセルを削除する。このモードでは、論理演算は行わずに消去するラインをすべて削除する。
HALFTONE	転送先の矩形のピクセルのブロックに転送元の矩形からピクセルを配置する。標準色は、転送元のピクセルの色に近い色で、転送先のピクセルのブロックを上塗りする。アプリケーションは伸縮モードにHALFTONEの設定後、ブラシの原点を設定するためSetBrushOrgEx関数を呼ばなければならない。それが失敗した場合、ブラシは現れない。
STRETCH_ANDSCANS	Windows 95では、BLACKONWHITEと同じ。
STRETCH_DELETESCANS	Windows 95では、COLORONCOLORと同じ。
STRETCH_HALFTONE	Windows 95では、HALFTONEと同じ。
STRETCH_ORSCANS	Windows 95では、WHITEONBLACKと同じ。
WHITEONBLACK	縮小化によってビットマップのラインが消去されるときに。OR演算子を用いてライン結合を行う。白黒のビットマップの場合、このモードでは、黒色のピクセルが消えて色付きまたは白色のピクセルが残る。

戻り値

正常終了	伸縮モードの値
異常終了	0

解説

SetStretchBltMode関数は、指定されたデバイスコンテキストのビットマップの伸縮モードを設定する。伸縮モードは、StretchBlt関数を使用して表示デバイス上にあるピクセル単位のビットマップの行や列をWindowsが結合する方法を定義するものである。

通常、BLACKONWHITE(STRETCH_ANDSCANS)モードおよびWHITEON BLACK(STRETCH_ORSCANS)モードは、モノクロビットマップ内の前景ピクセルを残すときに使う。また、COLORONCOLOR(STRETCH_DELETESCANS)モードは、カラービットマップ内の色を残すときに使われる。HALFTONEモードは、他の3種のモードより転送元イメージの加工をすることを要求する。高さや品質のイメージを加工するため他の処理より遅い。HALFTONEモードを設定された後で呼び出されるSetBrushOrgEx関数では、パフォーマンスに注意する。
追加された伸縮モードではパフォーマンスは、デバイスドライバの能力の信頼性を高める。

参　照　→SetBrushOrgEx,→StretchBlt

StretchBlt	gdi32.dll	95	NT
ビットマップの矩形領域コピー			

V C

```
BOOL StretchBlt(
        HDC     hdcDest,        //転送先のデバイスコンテキストのハンドル
        int     nXOriginDest,   //転送先の矩形の左上隅のx座標
        int     nYOriginDest,   //転送先の矩形の左上隅のy座標
        int     nWidthDest,     //転送先の矩形の幅
        int     nHeightDest,    //転送先の矩形の高さ
        HDC     hdcSrc,         //転送元のデバイスコンテキストのハンドル
        int     nXOriginSrc,    //転送元の矩形の左上隅のx座標
        int     nYOriginSrc,    //転送元の矩形の左上隅のy座標
        int     nWidthSrc,      //転送元の矩形の幅
        int     nHeightSrc,     //転送元の矩形の高さ
        DWORD   dwRop           //ラスタオペレーションコード
        );
```

V B

```
Declare Function StretchBlt Lib "gdi32" Alias "StretchBlt" (ByVal hdc As Long, ByVal x As Long, ByVal y As Long, ByVal nWidth As Long, ByVal nHeight As Long, ByVal hSrcDC As Long, ByVal xSrc As Long, ByVal ySrc As Long, ByVal nSrcWidth As Long, ByVal nSrcHeight As Long, By Val dwRop As Long) As Long
```

パラメータ

- *hdcDest*　転送先のデバイスコンテキストの識別子。
- *nXOriginDest*　転送先の矩形の左上隅の論理x座標を指定する。
- *nYOriginDest*　転送先の矩形の左上隅の論理y座標を指定する。
- *nWidthDest*　転送先の矩形の幅を論理単位で指定する。
- *nHeightDest*　転送先の矩形の高さを論理単位で指定する。
- *hdcSrc*　転送元のデバイスコンテキストの識別子。
- *nXOriginSrc*　転送元の矩形の左上隅の論理x座標を指定する。
- *nYOriginSrc*　転送元の矩形の左上隅の論理y座標を指定する。

- *nWidthSrc*　転送元の矩形の幅を論理単位で指定する。
- *nHeightSrc*　転送元の矩形の高さを論理単位で指定する。
- *dwRop*　実行するラスタオペレーションを指定する。ラスタオペレーションコードは、Windowsが現在のブラシ、転送元のビットマップ、転送先のビットマップを組み合わせて出力オペレーションの色を生成する方法を定義する。共通ラスタオペレーションコードは、BitBlt関数を参照。

戻り値

正常終了	TRUE
異常終了	FALSE

詳細なエラー情報は、GetLastError関数により取得。

解説

StretchBlt関数は、元の矩形領域から別の矩形領域へビットマップをコピーする。また、必要であれば、転送先の矩形に合うようにビットマップを拡大または縮小してコピーする。Windowsは、転送先のデバイスコンテキストの伸縮モード使用して、ビットマップを拡大または縮小する。

StretchBlt関数は、メモリ上で転送元のビットマップを拡大または縮小し、その結果を転送先の矩形にコピーする。パターンのカラーデータをこの結果とマージするときは、拡大された転送元のビットマップが転送先にコピーされるまでマージされない。

エンハンストメタファイルのときに、転送元デバイスコンテキストがエンハンストメタファイルのデバイスコンテキストを識別すると、エラーが発生する(関数がFALSEを返す)。

指定されたラスタオペレーションがブラシの場合、Windowsで使われるブラシは、転送先のデバイスコンテキスト内で選択されたブラシである。

転送先の座標は、転送先のデバイスコンテキストに合わせて変換される。転送元の座標は、転送元のデバイスコンテキストに合わせて変換される。

転送元の変換が、配置替えや分割であるとエラーが発生する。

転送先のビットマップ、転送元のビットマップ、パターンビットマップのカラー形式が異なる場合、StretchBlt関数は、転送先のビットマップに合わせて転送元のビットマップとパターンビットマップのカラー形式を変換する。

StretchBlt関数でモノクロのビットマップをカラーに変換するときには、白のビット(1)を背景色に、黒のビット(0)を前景色に設定する。カラーのビットマップをモノクロに変換するときには、背景色を白のビット(1)に、他の色を黒のビット(0)に設定する。この前景色と背景色には、転送先のデバイスコンテキストのものが使われる。

StretchBlt関数は、*nWidthSrc*と*nWidthDest*、または*nHeightSrc*と*nHeightDest*の各パラメータの符号が異なるときに、ビットマップのミラーイメージを作成する。*nWidthSrc*パラメータと*nWidthDest*パラメータの符号が異なるときは、x座標に沿ってビットマップのミラーイメージを作成する。*nHeightSrc*パラメータと*nHeightDest*パラメータの符号が異なるときは、y座標に沿ってビットマップのミラーイメージを作成する。すべてのデバイスでStretchBlt関数がサポートされているわけではない。GetDeviceCaps関数で詳しい情報を取得すること。

参照

→BitBlt,→GetDeviceCaps,MaskBlt,PlgBlt,→SetStretchBltMode

StretchDIBits	gdi32.dll	95	NT
ビットマップの矩形領域コピー			

V C

```
int StretchDIBits(
    HDC                  hdc,          //デバイスコンテキストのハンドル
    int                  XDest,        //転送先の左上隅の矩形のx座標
    int                  YDest,        //転送先の左上隅の矩形のy座標
    int                  nDestWidth,   //転送先の矩形の幅
    int                  nDestHeight,  //転送先の矩形の高さ
    int                  XSrc,         //転送元の左上隅の矩形のx座標
    int                  YSrc,         //転送元の左上隅の矩形のy座標
    int                  nSrcWidth,    //転送元の矩形の幅
    int                  nSrcHeight,   //転送元の矩形の高さ
    CONST VOID*          lpBits,       //ビットマップビットへのポインタ
    CONST BITMAPINFO*    lpBitsInfo,   //ビットマップデータへのポインタ
    UINT                 iUsage,       //使用法
    DWORD                dwRop         //ラスタオペレーションコード
    );
```

V B

Declare Function StretchDIBits Lib "gdi32" Alias "StretchDIBits" (ByVal hdc As Long, ByVal x As Long, ByVal y As Long, ByVal dx As Long, ByVal dy As Long, ByVal SrcX As Long, ByVal SrcY As Long, ByVal wSrcWidth As Long, ByVal wSrcHeight As Long, lpBits As Any, lpBitsInfo As BITMAPINFO, By Bal wUsage As Long, By Val dwRop As Long) As Long

パラメータ

- *hdc*　転送先のデバイスコンテキストの識別子。
- *XDest*　転送先の矩形の左上隅の論理x座標を指定する。
- *YDest*　転送先の矩形の左上隅の論理y座標を指定する。
- *nDestWidth*　転送先の矩形の幅を論理単位で指定する。
- *nDestHeight*　転送先の矩形の高さを論理単位で指定する。
- *XSrc*　DIB内の転送元の矩形のx座標をピクセル単位で指定する。
- *YSrc*　DIB内の転送元の矩形のy座標をピクセル単位で指定する。
- *nSrcWidth*　DIB内の転送元の矩形の幅をピクセル単位で指定する。
- *nSrcHeight*　DIB内の転送元の矩形の高さをピクセル単位で指定する。
- *lpBits*　バイト配列として格納されるDIBビットへのポインタ。
- *lpBitsInfo*　DIBに関する情報を含むBITMAPINFO構造体へのポインタ。

・*iUsage* BITMAPINFO構造体のbmiColorsメンバが用意されているか、赤、緑、青(RGB)の値が格納されているのかを指定する。このパラメータは、以下の値のいずれかになる。

値	意味
DIB_PAL_COLORS	カラーテーブルは、転送元のデバイスコンテキストの論理パレットへの16ビットのインデックスの配列である。
DIB_RGB_COLORS	カラーテーブルは、RGB値である。

・*dwRop* 転送元のピクセル、転送先のデバイスコンテキストのカレントブラシ、ピクセルを組み合わせて新しいイメージを生成する方法を指定する。指定できる値は、BitBlt関数の共通ラスタオペレーションコードを参照すること。

戻り値

正常終了	コピーされた走査線数
異常終了	GDI_ERROR

詳細なエラー情報は、GetLastError関数により取得。

解　説

StretchDIBits関数は、デバイスに依存しないビットマップ(DIB)の矩形のカラーデータを、転送先の矩形領域にコピーする。転送先の矩形が転送元の矩形より大きい場合は、転送先の矩形に合うようにカラーデータの高さや幅を拡大する。転送先の矩形が転送元の矩形より小さい場合は、指定されたラスタオペレーションで高さや幅を縮小する。

ボトムアップのDIBの原点は左下隅になる。トップダウンのDIBの原点は左上隅になる。StretchDIBits関数は、*nSrcWidth*と*nDestWidth*、または*nSrcHeight*と*nDestHeight*の各パラメータの符号が異なるときに、ビットマップのミラーイメージを作成する。*nSrcWidth*パラメータと*nDestWidth*パラメータの符号が異なるときは、x座標に沿ってビットマップのミラーイメージを作成する。*nSrcHeight*パラメータと*nDestHeight*パラメータの符号が異なるときは、y座標に沿ってビットマップのミラーイメージを作成する。

参　照

→SetMapMode,→SetStretchBltMode,→BITMAPINFO

Brush

CreateBrushIndirect

CreateBrushIndirect	gdi32.dll	95	NT
論理ブラシ作成			

V C

```
HBRUSH CreateBrushIndirect(
    CONST LOGBRUSH*    lplb    //ブラシ属性の構造体へのポインタ
    );
```

V B

```
Declare Function CreateBrushIndirect Lib "gdi32" Alias "CreateBrushIndirect" (lpLogBrush As LOGBRUSH) As Long
```

パラメータ

・*lplb*　　ブラシに関する情報を持つLOGBRUSH構造体のポインタ。

戻り値

正常終了	論理ブラシの識別子
異常終了	NULL

解説

CreateBrushIndirect関数は、指定されたスタイル、色、パターンを持つ論理ブラシを作成する。ブラシはWindowsが塗りつぶしに使うビットマップである。
アプリケーションはCreateBrushIndirect関数で作成したブラシをSelectObject関数を使うことで、どのようなデバイスコンテキストでも使用できる。
モノクロビットマップ(ピクセル当たり1つのプレーン、1つのビットを持つ)を使用して作成されたブラシは、カレントテキストの色と背景色を使用して描画される。ビットが0に設定されたピクセルはカレントテキストの色で描画され、ビットが1に設定されたピクセルは現在の背景色で描画される。
*lplb*により指されるLOGBRUSH構造体のlbStyleメンバがBS_PATTERNのとき、lbHatchメンバにより指されるビットマップは、DIBセクションにはならない。DIBセクションは、CreateDibSection関数により作成されたビットマップである。ビットマップが、DIBセクションであると、CreateBrushIndirect関数は失敗する。
論理ブラシが必要なくなったとき、DeleteObject関数を使用してブラシを削除しなければならない。

参照

CreateDibSection,DeleteObject,→GetBrushOrgEx,→LOGBRUSH,SelectObject,
→SetBrushOrgEx

CreateDIBPatternBrush	gdi32.dll	95	NT
論理ブラシ作成			

V C

```
HBRUSH CreateDIBPatternBrush(
    HGLOBAL  hglbDIBPacked, //デバイスに依存しないビットマップのハンドル
    UINT     fuColorSpec    //カラーテーブルデータ
    );
```

V B

Declare Function CreateDIBPatternBrush Lib "gdi32" Alias "CreateDIBPatternBrush" (ByVal hPackedDIB As Long, ByVal wUsage As Long) As Long

パラメータ

・*hglbDIBPacked*　デバイスに依存しないパックされたDIBを含むグローバルメモリオブジェクトの識別子。パックされたDIBは、BITMAPINFO構造体と、その直後に続くビットマップの各ピクセル数を定義するバイト配列で構成される。Windows 95では、ビットマップまたは、DIBから8×8ピクセルより大きなブラシを作成することはサポートしていない。大きなビットマップが指定された場合、ビットマップの一部だけが使用される。

・*fuColorSpec*　BITMAPINFO構造体のbmiColorsメンバが、赤、緑、青(RGB)の値と、現在実体化されている論理パレットへのインデックスのどちらを表すかを指定する。*fuColorSpec*パラメータには、以下の値のいずれかを指定する。

値	意味
DIB_PAL_COLORS	カラーテーブルは、選択されたブラシのデバイスコンテキストの論理パレットへの16ビットインデックスの配列。
DIB_RGB_COLORS	カラーテーブルは、RGB値である。

戻り値

正常終了	論理ブラシのハンドル
異常終了	NULL

解説

CreateDIBPatternBrush関数は、デバイスに依存しないビットマップ(DIB)により指定されたパターンを持つ、論理ブラシを作成する。このブラシは、ラスタオペレーションをサポートするすべてのデバイスで選択できる。

この関数は、Windows3.0以前のバージョンで記述されたアプリケーションとの互換性のために用意されている。Win32ベースのアプリケーションでは、CreateDIBPatternBrushPt関数を使用する。

アプリケーションがモノクロデバイスコンテキストの中に2色のDIBパターンブラシを選択した場合、DIBで指定される色は無視され、現在の背景色とデバイスコンテキストの前景色によるパターンブラシが表示される。DIBの最初の色(DIBカラーテーブルのオフセット0)に割り当てられるピクセルは、前景色を使用して表示され、2番目の色(カラーテーブルのオフセット1)に割り当てられるピクセルは、背景色を使用して表示される。

論理ブラシが必要なくなったとき、DeleteObject関数を使用してブラシを削除しなければならない。

参　照

→BITMAPINFO,→CreateDIBPatternBrushPt,→CreateHatchBrush,
→CreatePatternBrush,→CreateSolidBrush,DeleteObject,SetBkColor,→SetTextColor

CreateDIBPatternBrushPt	gdi32.dll	95	NT
論理ブラシ作成			

V　C

```
HBRUSH CreateDIBPatternBrushPt(
    CONSTVOID*  lpPackedDIB,  //構造体とビットマップビットのポインタ
    UINT        iUsage        //使用法フラグ
    );
```

V　B

```
Declare Function CreateDIBPatternBrushPt Lib "gdi32" Alias "CreateDIBPattern
BrushPt" (lpPackedDIB As Any, ByVal iUsage As Long) As Long
```

パラメータ

・*lpPackedDIB*　BITMAPINFO構造体と、その直後に続くビットマップの各ピクセル数を定義するバイト配列で構成されるDIBへのポインタ。Windows 95では、ビットマップまたはDIBから8×8ピクセルより大きなブラシを作成することはサポートしていない。大きなビットマップが指定された場合、ビットマップの一部だけが使用される。

・*iUsage*　BITMAPINFO構造体のbmiColorsメンバが、赤、緑、青(RGB)の値と、現在実体化されている論理パレットへのインデックスのどちらを表すかを指定する。*iUsage*パラメータには、以下の値のいずれかを指定する。

値	意味
DIB_PAL_COLORS	カラーテーブルは、選択されたブラシのデバイスコンテキストの論理パレットへの16ビットインデックスの配列。
DIB_RGB_COLORS	カラーテーブルは、RGB値である。

戻り値

正常終了	論理ブラシの識別子
異常終了	NULL

解　説

CreateDIBPatternBrushPt関数は、デバイスに依存しないビットマップ(DIB)により指定されたパターンを持つ論理ブラシを作成する。
ブラシはWindowsが塗りつぶしに使うビットマップである。
アプリケーションはCreateDIBPatternBrushPt関数で作成したブラシをSelectObject関数を使うことでどのようなデバイスコンテキストでも使用できる。
論理ブラシが必要なくなったときは、DeleteObject関数を使用してブラシを削除する。

参　照

→BITMAPINFO,→CreateDIBPatternBrush,→CreateHatchBrush,→CreatePatternBrush,
→CreateSolidBrush,DeleteObject,→GetBrushOrgEx,SelectObject,→SetBrushOrgEx

CreateHatchBrush	gdi32.dll	95	NT
論理ブラシ作成			

V C

```
HBRUSH CreateHatchBrush(
    int          fnStyle,     //ハッチスタイル
    COLORREF     clrref       //色の値
    );
```

V B

Declare Function CreateHatchBrush Lib "gdi32" Alias "CreateHatchBrush" (ByVal nIndex As Long, ByVal crColor As Long) As Long

パラメータ

・*fnStyle*　　ブラシのハッチスタイルを指定する。このパラメータは、以下の値のいずれかである。

値	意味
HS_BDIAGONAL	45度上向きの左から右へのハッチ
HS_CROSS	水平と垂直のクロスハッチ
HS_DIAGCROSS	45度のクロスハッチ
HS_FDIAGONAL	45度下向きの左から右へのハッチ
HS_HORIZONTAL	水平ハッチ
HS_VERTICAL	垂直ハッチ

・*clrref*　　ハッチに使うブラシの前景色を指定する。

戻り値

正常終了	論理ブラシのハンドル
異常終了	NULL

解　説

CreateHatchBrush関数は、指定されたハッチパターンと色を持つ論理ブラシを作成する。ブラシはWindowsが塗りつぶしに使うビットマップである。
アプリケーションはCreateHatchBrush関数で作成したブラシをSelectObject関数を使うことでどのようなデバイスコンテキストでも使用できる。
アプリケーションが調和した色で親ウィンドウと子ウィンドウの背景色を塗りつぶすためにハッチブラシを使うとき、子ウィンドウの背景色を塗る前にブラシの原点を設定する必要がある。これは、アプリケーションでSetBrushOrgEx関数を呼び出すことにより行われる。アプリケーションは、SetBrushOrgEx関数を呼び出すことによりカレントブラシの原点を取得できる。ブラシが必要なくなったときは、DeleteObject関数により削除する。

参　照

→CreateDIBPatternBrush,→CreateDIBPatternBrushPt,→CreatePatternBrush,
→CreateSolidBrush,DeleteObject,→GetBrushOrgEx,SelectObject,→SetBrushOrgEx

CreatePatternBrush	gdi32.dll	95	NT
論理ブラシ作成			

V　C

```
HBRUSH CreatePatternBrush(
    HBITMAP     hbmp     //ビットマップのハンドル
    );
```

V　B

Declare Function CreatePatternBrush Lib "gdi32" Alias "CreatePatternBrush" (ByVal hBitmap As Long) As Long

パラメータ

・*hbmp*　論理ブラシを作成するために使用するビットマップの識別子。Windows 95では、ビットマップまたはDIBから8×8ピクセルより大きなブラシを作成することはサポートしていない。大きなビットマップが指定された場合、ビットマップの一部だけが使用される。

戻り値

正常終了	論理ブラシの識別子
異常終了	NULL

解　説

CreatePatternBrush関数は、指定されたビットマップパターンを持つ論理ブラシを作成する。このビットマップは、CreateDIBSection関数で作成されたDIBではならない。ブラシはWindowsが塗りつぶしに使うビットマップである。
アプリケーションはCreatePatternBrush関数で作成したブラシをSelectObject関数を使用することでどのようなデバイスコンテキストでも使用できる。
DeleteObject関数を使用して関連するビットマップに対して影響を与えずにパターンブラシを削除できる。つまり、同じビットマップを使用してパターンブラシをいくつでも作成することができる。
モノクロビットマップ(ピクセル当たり1つのプレーン、1つのビットを持つ)を使用して作成されたブラシは、現在のテキストの色と背景色を使用して描画される。ビットが0に設定されたピクセルはカレントのテキストの色で描画され、ビットが1に設定されたピクセルはカレントの背景色で描画される。
*hbmp*により識別されるビットマップは、CreateDIBSection関数により作成されたビットマップであるDIBセクションではない。ビットマップが、DIBセクションであった場合は、CreatePatternBrush関数は失敗する。

参　照

→CreateBitmap,CreateBitmapIndirect,→CreateCompatibleBitmap,
→CreateDIBPatternBrush,→CreateDIBPatternBrushPt,CreateDIBSection,
→CreateHatchBrush,DeleteObject,→GetBrushOrgEx,→LoadBitmap,
SelectObject,→SetBrushOrgEx

CreateSolidBrush	gdi32.dll	95	NT
論理ブラシ作成			

V C

```
HBRUSH CreateSolidBrush(
    COLORREF    crColor    //ブラシの色
    );
```

V B

```
Declare Function CreateSolidBrush Lib "gdi32" Alias "CreateSolidBrush" (ByVal crColor As Long) As Long
```

パラメータ

・*crColor*　　ブラシの色を指定する。

戻り値

正常終了	論理ブラシの識別子
異常終了	NULL

解　説

CreateSolidBrush関数は、指定された純色(solidcolor)のブラシを作成する。作成されたブラシは任意のデバイスのカレントブラシとして選択できる。ブラシはWindowsが塗りつぶしに使うビットマップである。
アプリケーションはCreateSolidBrush関数で作成したブラシをSelectObject関数を使うことでどのようなデバイスコンテキストでも使用できる。

参　照

→CreateDIBPatternBrush,→CreateDIBPatternBrushPt,→CreateHatchBrush,
→CreatePatternBrush,DeleteObject,SelectObject

GetBrushOrgEx	gdi32.dll	95	NT
カレントブラシの原点取得			

V　C

```
BOOL GetBrushOrgEx(
    HDC         hdc,     //デバイスコンテキストのハンドル
    LPPOINT     lppt     //原点の構造体へのアドレス
    );
```

V　B

Declare Function GetBrushOrgEx Lib "gdi32" Alias "GetBrushOrgEx" (ByVal hDC As Long, lpPoint As POINTAPI) As Boolean

パラメータ

- *hdc*　　デバイスコンテキストの識別子。
- *lppt*　　ブラシの原点のデバイス座標が返されるPOINT構造体へのポインタ。

戻り値

正常終了	TRUE
異常終了	FALSE

詳細なエラー情報は、GetLastError関数により取得。

解　説

GetBrushOrgEx関数は、指定されたデバイスコンテキストのカレントブラシの原点を取得する。この関数は、GetBrushOrg関数に代わるものである。ブラシはWindowsが塗りつぶしに使うビットマップである。ブラシの原点は、ビットマップの中の1ピクセルの位置を指定する0から7の間の値の座標である。デフォルトのブラシの原点は、座標(0,0)である。水平方向の座標で、値0はピクセルの最も左に位置し、値7は最も右に位置する。垂直方向の座標で、値0はピクセルの最も上に位置し、値7は最も下に位置する。Windowsは、どのような描画操作においてもクライアント領域の中にブラシの原点を配置して開始位置を決定する。例えば、原点を(2,3)に設定すると、Windowsはクライアント領域のブラシの(0,0)を原点として(2,3)の位置に配置する。
アプリケーションが調和した色で親ウィンドウと子ウィンドウの背景色を塗りつぶすためにブラシを使うとき、子ウィンドウの背景色を描画する前に親ウィンドウを描画し、ブラシの原点を設定する必要がある。
WindowsNTでは、オペレーティングシステムが、ウィンドウ操作するデバイスコンテキストの原点を自動的にトレースし、表面のパターン列を維持するのに必要なブラシを適応させる。
Windows 95では、ブラシの原点の自動的なトレースはサポートされていない。アプリケーションは、ブラシを使う前にUnrealizeObject、SetBrushOrgEx、SelectObject関数を使わなければならない。

参　照

→POINT,SelectObject,→SetBrushOrgEx,→UnrealizeObject

GetSysColorBrush	user32.dll	95	NT
論理ブラシハンドル取得			

V C

```
HBRUSH GetSysColorBrush(
    int        nIndex       //システムカラーインデックス
    );
```

V B

Declare Function GetSysColorBrush Lib "user32" Alias "GetSysColorBrush" (ByVal nIndex As Long) As Long

パラメータ

・*nIndex*　カラーインデックスを指定する。この値は、21個あるウィンドウの構成要素(アクティブ/非アクティブウィンドウのタイトルバーの色など)の1つを描くために使う色と同じである。

戻り値

正常終了	論理ブラシの識別子
異常終了	NULL

詳細なエラー情報は、GetLastError関数により取得。

解　説

GetSysColorBrush関数は、カラーインデックスを指定するために対応する論理ブラシのハンドルを検索する。ブラシは、Windowsが塗りつぶされた図形の内側を描画するために使うビットマップである。アプリケーションは、GetSysColor関数を呼び出すことにより、カレントシステムカラーを検索できる。また、アプリケーションはSetSysColors関数を呼び出すことにより、カレントシステムカラーを設定することができる。

アプリケーションは、ウィンドウがシステムブラシを使うためウィンドウクラスを登録してはならない。

参　照

GetSysColor,SetSysColors

SetBrushOrgEx	gdi32.dll	95	NT
ブラシ原点の設定			

V C

```
BOOL SetBrushOrgEx(
    HDC        hdc,         //デバイスコンテキストのハンドル
    int        nXOrg,       //新しい原点のx座標
    int        nYOrg,       //新しい原点のy座標
    LPPOINT    lppt         //以前のブラシの原点を指定する構造体へのポイ
                              ンタ
    );
```

V B

Declare Function SetBrushOrgEx Lib "gdi32" Alias "SetBrushOrgEx" (ByVal hdc As Long, ByVal nXOrg As Long, ByVal nYOrg As Long, lppt As POINTAPI) As Long

パラメータ

- *hdc*　デバイスコンテキストの識別子。
- *nXOrg*　デバイス単位で、新しいブラシの原点のx座標を指定する。この値は、0から7の範囲内である。
- *nYOrg*　デバイス単位で、新しいブラシの原点のy座標を指定する。この値は、0から7の範囲内である。
- *lppt*　以前のブラシの原点を受け取るPOINT構造体へのポインタ。以前のブラシの原点が必要でないときは、このパラメータはNULLでかまわない。

戻り値

正常終了	TRUE
異常終了	FALSE

詳細なエラー情報は、GetLastError関数で取得。

解　説

SetBrushOrgEx関数は、指定されたデバイスコンテキストよりアプリケーションが選択したブラシのためにGDIが割り当てたブラシの原点を設定する。この関数は、SetBrushOrg関数に代わるものである。ブラシはオペレーティングシステムが塗りつぶしに使うビットマップである。ブラシの原点は、ビットマップの中の1ピクセルの位置を指定する0から7の間の値の座標である。デフォルトのブラシの原点は、座標(0,0)である。水平方向の座標で、値0はピクセルの最も左に位置し、値7は最も右に位置する。垂直方向の座標で、値0はピクセルの最も上に位置し、値7は最も下に位置する。

Windowsは、どのような描画操作のブラシの開始位置を決定するときも、指定されるクライアント領域の中の位置へブラシの原点を配置する。たとえば、原点を(2,3)に設定すると、Windowsはクライアント領域のブラシの(0,0)を原点として(2,3)の位置に配置する。

オペレーティングシステムが、ウィンドウ操作するデバイスコンテキストの原点を自動的にトレースし、表面のパターン列を維持するのに必要なブラシを適応させる。このブラシの原点は、クライアントエリアの左上隅に設定される。

アプリケーションは、SetStretchBltMode関数によりビットマップをHALFTONEモードに設定した後で、SetBrushOrgEx関数を呼び出すべきである。

WindowsNTでは、オペレーティングシステムが、ウィンドウ操作するデバイスコンテキストの原点を自動的にトレースし、表面のパターン列を維持するのに必要なブラシを適応させる。

Windows 95では、ブラシの原点の自動的なトレースはサポートされていない。アプリケーションは、ブラシを利用する前にUnrealizeObject、SetBrushOrgEx、SelectObject関数を使わなければならない。

参　照

→GetBrushOrgEx,→POINT,SelectObject,→SetStretchBltMode,→UnrealizeObject

Clipping

ExcludeClipRect

ExcludeClipRect	gdi32.dll	95	NT

既存のクリッピングリージョンから新規リージョン作成

V C

```
int ExcludeClipRect(
    HDC     hdc,          //デバイスコンテキストのハンドル
    int     nLeftRect,    //矩形の左上隅のx座標
    int     nTopRect,     //矩形の左上隅のy座標
    int     nRightRect,   //矩形の右下隅のx座標
    int     nBottomRect   //矩形の右下隅のy座標
    );
```

V B

```
Declare Function ExcludeClipRect Lib "gdi32" Alias "ExcludeClipRect" (ByVal hdc As Long, ByVal X1 As Long, ByVal Y1 As Long, ByVal X2 As Long, ByVal Y2 As Long) As Long
```

パラメータ

- *hdc* デバイスコンテキストの識別子。
- *nLeftRect* 矩形の左上隅の論理x座標を指定する。
- *nTopRect* 矩形の左上隅の論理y座標を指定する。
- *nRightRect* 矩形の右下隅の論理x座標を指定する。
- *nBottomRect* 矩形の右下隅の論理y座標を指定する。

戻り値

関数が正常に終了した場合は、新しいクリッピングリージョンのタイプを示す値を返す。戻り値は、以下の値のうち1つである。

NULLREGION	リージョンは空。
SIMPLEREGION	リージョンは、単一の矩形である。
COMPLEXREGION	リージョンは、複数の矩形である。
ERROR	リージョンは、作成されない。

解　説

ExcludeClipRect関数は、既存のクリッピングリージョンから指定された矩形の領域を除いて、新しいクリッピングリージョンを作成する。
指定された矩形の右下隅は、クリッピングリージョンから出ない。

参　照

→IntersectClipRect

ExtSelectClipRgn	gdi32.dll	95	NT
クリッピングリージョンの結合			

V　C

```
int ExtSelectClipRgn(
    HDC     hdc,     //デバイスコンテキストのハンドル
    HRGN    hrgn,    //リージョンのハンドル
    int     fnMode   //リージョン選択モード
    );
```

V　B

```
Declare Function ExtSelectClipRgn Lib "gdi32" Alias "ExtSelectClipRgn" (ByVal hdc As Long, ByVal hRgn As Long, ByVal fnMode As Long) As Long
```

パラメータ

- *hdc*　デバイスコンテキストの識別子。
- *hrgn*　選択されたリージョンの識別子。*fnMode*にRGN_COPYを指定した場合、このハンドルはNULLである。
- *fnMode*　実行される操作を指定する。以下の値のうち1つを指定する。

値	意味
RGN_AND	新しいクリッピングリージョンは、カレントクリッピングリージョンの領域に重なる。そのリージョンは、*hrgn*で識別される。
RGN_COPY	新しいクリッピングリージョンは、*hrgn*で識別されるリージョンのコピーである。これは、SelectClipRgnと同じである。*hrgn*により識別されたリージョンがNULLのとき、新しいクリッピングリージョンは、デフォルトのクリッピングリージョンとなる(デフォルトのクリッピングリージョンは、空のリージョンである)。
RGN_DIFF	新しいクリッピングリージョンは、*hrgn*により識別されたリージョンに含まれないカレントのクリッピングリージョンの領域と結合する。
RGN_OR	新しいクリッピングリージョンは、カレントのクリッピングリージョンと結合する。そのリージョンは、*hrgn*により識別される。
RGN_XOR	新しいクリッピングリージョンは、カレントのクリッピングリージョンと結合する。そのリージョンは、*hrgn*により識別されるが、重なった領域は含まない。

戻り値

関数が正常に終了した場合は、新しいクリッピングリージョンのタイプを示す値を返す。戻り値は、以下の値のうち1つである。

NULLREGION	リージョンは空。
SIMPLEREGION	リージョンは、単一の矩形である。
COMPLEXREGION	リージョンは、複数の矩形である。
ERROR	エラーが発生した。

解　説

ExtSelectClipRgn関数は指定されたモードで、カレントのクリッピングリージョンと指定されたリージョンを結合する。
この関数が呼び出されてエラーが発生したとしても、指定されたデバイスにおける以前のクリッピングリージョンに影響はない。
*hrgn*パラメータで識別されるリージョンのコピーだけが使用される。呼び出された後、このリージョン自身は再度使用されるか削除される。

参　照

→SelectClipRgn

GetClipBox	gdi32.dll	95	NT
矩形サイズの取得			

V　C

```
int GetClipBox(
        HDC         hdc,      //デバイスコンテキストのハンドル
        LPRECT      lprc      //矩形の構造体のアドレス
        );
```

V　B

```
Declare Function GetClipBox Lib "gdi32" Alias "GetClipBox" (ByVal hdc As Long,
lpRect As RECT) As Long
```

パラメータ

- *hdc*　　デバイスコンテキストの識別子。
- *lprc*　　矩形の寸法を受け取るRECT構造体へのポインタ。

戻り値

関数が正常に終了した場合は、クリッピングボックスのタイプを示す値を返す。戻り値は、以下の値のうち1つである。

NULLREGION	リージョンは空。
SIMPLEREGION	リージョンは、単一の矩形である。
COMPLEXREGION	リージョンは、複数の矩形である。
ERROR	エラーが発生した。

GetClipBox関数は、デバイスコンテキストに基づいた論理座標を返す。

解　説

GetClipBox関数は、デバイス上の目に見える領域を囲む最小の矩形の寸法を取得する。目に見える領域は、カレントクリッピングリージョンやクリップパスや重なっているウィンドウにより定義される。

参　照

→RECT

GetClipRgn	gdi32.dll	95	NT
クリッピングリージョン識別用のハンドル取得			

V　C

```
int GetClipRgn(
        HDC          hdc,      //デバイスコンテキストのハンドル
        HRGN         hrgn      //リージョンのハンドル
        );
```

V　B

Declare Function GetClipRgn Lib "gdi32" Alias "GetClipRgn" (ByVal hdc As Long, ByVal hRgn As Long) As Long

パラメータ

- *hdc*　デバイスコンテキストの識別子。
- *hrgn*　この関数が呼び出される以前に存在するリージョンを識別する。関数が戻った後、このパラメータはカレントクリッピングリージョンのコピーを識別する。

戻り値

正常終了　クリッピングリージョンが存在しない場合	0
正常終了　クリッピングリージョンが存在する場合	1
異常終了	-1

解　説

GetClipRgn関数は、指定されたデバイスコンテキストのカレントアプリケーション定義のクリッピングリージョンを識別するハンドルを取得する。
アプリケーション定義のクリッピングリージョンは、SelectClipRgn関数により識別されるクリッピングリージョンである。BeginPaint関数により作成されたクリッピングリージョンではない。
関数が正常に終了した場合、*hrgn*パラメータはカレントクリッピングリージョンのコピーを識別する。このコピーの変更はカレントクリッピングリージョンには影響しない。

参　照

→BeginPaint,→SelectClipRgn

GetMetaRgn	gdi32.dll	95	NT
カレントメタリージョンの取得			

V　C

```
int GetMetaRgn(
        HDC             hdc,        //デバイスコンテキストのハンドル
        HRGN            hrgn        //リージョンのハンドル
        );
```

V　B

Declare Function GetMetaRgn Lib "gdi32" Alias "GetMetaRgn" (ByVal hdc As Long, ByVal hRgn As Long) As Long

パラメータ

- *hdc*　デバイスコンテキストの識別子。
- *hrgn*　この関数が呼び出される以前に存在するリージョンを識別する。関数が戻った後、このパラメータは、カレントメタリージョンのコピーを識別する。

戻り値

正常終了	1
異常終了	0

解　説

GetMetaRgn関数は、指定されたデバイスコンテキストのカレントメタリージョンを取得する。
関数が正常に終了した場合、*hrgn*パラメータはカレントメタリージョンのコピーを識別する。このコピーの変更は、カレントメタリージョンには影響しない。
デバイスコンテキストのカレントクリッピングリージョンは、クリッピングリージョンとメタリージョンの交点により定義される。

参　照

→SetMetaRgn

IntersectClipRect	gdi32.dll	95	NT
指定矩形とクリッピングリージョンから新規クリッピングリージョンの作成			

V　C

```
int IntersectClipRect(
    HDC   hdc,          //デバイスコンテキストのハンドル
    int   nLeftRect,    //矩形の左上隅のx座標
    int   nTopRect,     //矩形の左上隅のy座標
    int   nRightRect,   //矩形の右下隅のx座標
    int   nBottomRect   //矩形の右下隅のy座標
    );
```

V　B

```
Declare Function IntersectClipRect Lib "gdi32" Alias "IntersectClipRect" (ByVal hdc As Long, ByVal X1 As Long, ByVal Y1 As Long, ByVal X2 As Long, ByVal Y2 As Long) As Long
```

パラメータ

- *hdc*　デバイスコンテキストの識別子。
- *nLeftRect*　矩形の左上隅の論理x座標を指定する。
- *nTopRect*　矩形の左上隅の論理y座標を指定する。
- *nRightRect*　矩形の右下隅の論理x座標を指定する。
- *nBottomRect*　矩形の右下隅の論理y座標を指定する。

戻り値

関数が正常に終了した場合は、新しいクリッピングリージョンのタイプを示す値を返す。戻り値は、以下の値のうち1つである。

正常終了	新規クリッピングリージョンのタイプを示す値。
NULLREGION	リージョンは空。
SIMPLEREGION	リージョンは、単一の矩形である。
COMPLEXREGION	リージョンは、複数の矩形である。
ERROR	エラーが発生した(カレントクリッピングリージョンには影響しない)。

解　説

IntersectClipRect関数は、指定された矩形とカレントクリッピングリージョンの交差する部分から、新しいクリッピングリージョンを作成する。
与えられた矩形の最も右下隅は、クリッピングリージョンに含まれない。

参　照

→ExcludeClipRect

OffsetClipRgn	gdi32.dll	95	NT
クリッピングリージョンの移動			

V C

```
int OffsetClipRgn(
    HDC     hdc,        //デバイスコンテキストのハンドル
    int     nXOffset,   //x軸方向のオフセット
    int     nYOffset    //y軸方向のオフセット
    );
```

V B

```
Declare Function OffsetClipRgn Lib "gdi32" Alias "OffsetClipRgn" (ByVal hdc As Long, ByVal x As Long, ByVal y As Long) As Long
```

パラメータ

- *hdc* デバイスコンテキストの識別子。
- *nXOffset* 左右に移動する論理単位の数値を指定する。
- *nYOffset* 上下に移動する論理単位の数値を指定する。

戻り値

関数が正常に終了した場合は、新しいクリッピングリージョンのタイプを示す値を返す。戻り値は、以下の値のうち1つである。

正常終了	新規クリッピングリージョンのタイプを示す値
NULLREGION	リージョンは空。
SIMPLEREGION	リージョンは、単一の矩形である。
COMPLEXREGION	リージョンは、複数の矩形である。
ERROR	エラーが発生した(カレントクリッピングリージョンには影響しない)。

解 説

OffsetClipRgn関数は、デバイスコンテキストのクリッピングリージョンを、指定されたオフセットだけ移動する。

参 照

→SelectClipRgn

PtVisible	gdi32.dll	95	NT
指定点とクリッピングリージョンの位置関係			

V　C

```
BOOL PtVisible(
        HDC        hdc,     //デバイスコンテキストのハンドル
        int        nX,      //点のx座標
        int        nY       //点のy座標
        );
```

V　B

Declare Function PtVisible Lib "gdi32" Alias "PtVisible" (ByVal hdc As Long, ByVal x As Long, ByVal y As Long) As Long

パラメータ

- *hdc*　　デバイスコンテキストの識別子。
- *nX*　　点の論理x座標を指定する。
- *nY*　　点の論理y座標を指定する。

戻り値

クリッピングリージョン内に点が位置する。	TRUE
クリッピングリージョン内に点は位置しない。	FALSE

解　説

PtVisible関数は、指定された点がデバイスコンテキストのクリッピングリージョン内にあるかどうかを調べる。

参　照

→RectVisible

RectVisible	gdi32.dll	95	NT
矩形領域とクリッピングリージョンの位置関係			

V　C

```
BOOL RectVisible(
        HDC            hdc,     //デバイスコンテキストのハンドル
        CONST RECT*    lprc     //矩形構造体へのポインタ
        );
```

V　B

Declare Function RectVisible Lib "gdi32" Alias "RectVisible" (ByVal hdc As Long, lpRect As RECT) As Long

パラメータ

- *hdc*　　デバイスコンテキストの識別子。
- *lprc*　　指定された矩形の論理座標を格納するRECT構造体へのポインタ。

戻り値

クリッピングリージョン内に矩形が一部でも含まれる	TRUE
クリッピングリージョン内に矩形は含まれない	FALSE

解　説

RectVisible関数は、指定された矩形の任意の部分がデバイスコンテキストのクリッピングリージョン内にあるかどうかを調べる。

参　照

CreateRectRgn,→PtVisible,→RECT,→SelectClipRgn

SelectClipPath	gdi32.dll	95	NT
カレントパスをクリッピングリージョンに選択			

VC

```
BOOL SelectClipPath(
    HDC      hdc,     //デバイスコンテキストのハンドル
    int      iMode    //クリッピングモード
    );
```

VB

Declare Function SelectClipPath Lib "gdi32" Alias "SelectClipPath" (ByVal hdc As Long, ByVal iMode As Long) As Long

パラメータ

- *hdc* パスのデバイスコンテキストの識別子。
- *iMode* パスの使い方を指定する。以下の値を設定する。

値	意味
RGN_AND	新しいクリッピングリージョンは、カレントクリッピングリージョン(重なる領域)の交点とカレントパスを含む。
RGN_COPY	新しいクリッピングリージョンは、カレントパスである。
RGN_DIFF	新しいクリッピングリージョンは、カレントパスに含まれないカレントのクリッピングリージョンの領域を含む。
RGN_OR	新しいクリッピングリージョンは、カレントのクリッピングリージョンの集合(結合された領域)とカレントパスを含む。
RGN_XOR	新しいクリッピングリージョンは、カレントのクリッピングリージョンの集合とカレントパスを含む。ただし、重なった領域は含まない。

戻り値

正常終了	TRUE
異常終了	FALSE

詳細なエラー情報は、GetLastError関数により取得。GetLastError関数は、以下のエラーコードを返す。

ERROR_CAN_NOT_COMPLETE,ERROR_INVALID_PARAMETER,
ERROR_NOT_ ENOUGH_MEMORY

解説

SelectClipPath関数は、カレントパスをデバイスコンテキストのクリッピングリージョンとして選択し、指定されたモードを使用して新しいリージョンを結合する。
*hdc*パラメータにより識別されるデバイスコンテキストは、結合するパスを含まなければならない。

参照

BeginPath,EndPath

SelectClipRgn	gdi32.dll	95	NT
リージョンをカレントクリッピングリージョンに選択			

V　C

```
int SelectClipRgn(
        HDC     hdc,     //デバイスコンテキストのハンドル
        HRGN    hrgn     //選択されたリージョンのハンドル
        );
```

V　B

```
Declare Function SelectClipRgn Lib "gdi32" Alias "SelectClipRgn" (ByVal hdc As Long, ByVal hRgn As Long) As Long
```

パラメータ

・*hdc*　デバイスコンテキストの識別子。
・*hrgn*　選択するリージョンの識別子。

戻り値

関数が正常に終了した場合は、リージョンのタイプを示す値を返す。戻り値は、以下の値のうち1つである。

NULLREGION	リージョンは空。
SIMPLEREGION	リージョンは、単一の矩形である。
COMPLEXREGION	リージョンは、複数の矩形である。
ERROR	エラーが発生した(以前のクリッピングリージョンには影響しない)。

解　説

SelectClipRgn関数は、リージョンを、指定されたデバイスコンテキストのカレントクリッピングリージョンとして選択する。
選択されたリージョンのコピーだけ使用される。リージョン自身は、別のデバイスコンテキストの任意の番号を選択され削除される。
SelectClipRgn関数は、デバイス単位で指定されるリージョンの座標を仮定する。
デバイスコンテキストのクリッピングリージョンを移動するには、NULLのリージョンハンドルを指定する。

参　照

→ExtSelectClipRgn

SetMetaRgn	gdi32.dll	95	NT
現在のメタリージョンとクリッピングリージョンから新規のメタリージョンを作成			

V C

```
int SetMetaRgn(
        HDC          hdc        //デバイスコンテキストのハンドル
        );
```

V B

Declare Function SetMetaRgn Lib "gdi32" Alias "SetMetaRgn" (ByVal hdc As Long) As Long

パラメータ

・*hdc*　デバイスコンテキストの識別子。

戻り値

関数が正常に終了した場合は、新しいクリッピングリージョンのタイプを示す値を返す。戻り値は、以下の値のうち1つである。

NULLREGION	リージョンは空。
SIMPLEREGION	リージョンは、単一の矩形である。
COMPLEXREGION	リージョンは、複数の矩形である。
ERROR	エラーが発生した(以前のクリッピングリージョンには影響しない)。

解　説

SetMetaRgn関数は、カレントメタリージョンとともにデバイスコンテキストを指定するカレントクリッピングリージョンを結合し、指定されたデバイスコンテキストのために新しいメタリージョンとして結合されたリージョンを格納する。クリッピングリージョンは、NULLのリージョンとして再設定する。
デバイスコンテキストのカレントクリッピングリージョンは、クリッピングリージョンとメタリージョンの結合により定義される。
SetMetaRgn関数は、アプリケーションのオリジナルデバイスコンテキストが、SaveDC関数により格納された後で呼び出される。

参　照

→GetMetaRgn,SaveDC

Color

AnimatePalette	gdi32.dll	95	NT
論理パレットエントリの置き換え			

V C

```
BOOL AnimatePalette(
    HPALETTE              hpal,        //論理カラーパレットのハンドル
    UINT                  iStartIndex, //最初の論理パレットエントリ
    UINT                  cEntries,    //論理パレットのエントリ数
    CONST PALETTEENTRY    *ppe         //最初の置換物へのポインタ
    );
```

V B

```
Declare Function AnimatePalette Lib "gdi32" Alias "AnimatePaletteA" (ByVal hPalette As Long, ByVal wStartIndex As Long, ByVal wNumEntries As Long, lpPaletteColors As PALETTEENTRY) As Long
```

パラメータ

- *hpal*　論理パレットの識別子。
- *iStartIndex*　置き換える最初の論理パレットのエントリを指定する。
- *cEntries*　置き換えるエントリ数を指定する。
- *ppe*　カレントエントリを置き換えるために使われるPALETTEENTRY構造体の配列の最初のメンバへのポインタ。

戻り値

正常終了	TRUE
異常終了	FALSE

詳細なエラー情報は、GetLastError関数により取得。

解　説

AnimatePalette関数は、指定された論理パレットのエントリを置き換える。
アプリケーションは、GetDeviceCaps関数を呼び出しRASTERCAPS定数を指定することにより、パレット操作をサポートするデバイスを決めることができる。
AnimatePalette関数は、LOGPALETTE構造体のpalPalEntryメンバにPC_RESERVEDフラグがセットされているときだけエントリを変更できる。
与えられたパレットがアクティブウィンドウと結合すると、パレットの色はすぐに変更される。

参　照

→CreatePalette,→GetDeviceCaps,→LOGPALETTE,→PALETTEENTRY

CreateHalftonePalette	gdi32.dll	95	NT
ハーフトーンパレットの作成			

V C

```
HPALETTE CreateHalftonePalette(
    HDC          hdc       //デバイスコンテキストのハンドル
    );
```

V B

Declare Function CreateHalftonePalette Lib "gdi32" Alias "CreateHalftonePalette" (ByVal hdc As Long) As Long

パラメータ

・*hdc*　　デバイスコンテキストの識別子。

戻り値

正常終了	論理ハーフトーンパレットの識別子
異常終了	0

詳細なエラー情報は、GetLastError関数により取得。

解 説

CreateHalftonePalette関数は、指定されたデバイスコンテキストのハーフトーンパレットを作成する。
アプリケーションが、デバイスコンテキストの伸縮モードにHALFTONEを設定することにより、ハーフトーンパレットを作成する。StretchBlt関数やStretchDIBits関数が呼び出される前に、CreateHalftonePalette関数により取得した論理ハーフトーンパレットを、デバイスコンテキストより選択し実現させる。
必要なくなったパレットは、DeleteObject関数により削除する。

参 照

DeleteObject,→RealizePalette,→SelectPalette,→SetStretchBltMode,→StretchDIBits,
→StretchBlt

CreatePalette	gdi32.dll	95	NT
論理カラーパレットの作成			

V C

```
HPALETTE CreatePalette(
    CONST LOGPALETTE    *lplgpl //論理カラーパレットへのポインタ
    );
```

V B

Declare Function CreatePalette Lib "gdi32" Alias "CreatePalette" (lpLogPalette As LOGPALETTE) As Long

パラメータ

・*lplgpl*　論理カラーパレット内の色に関する情報を格納する、LOGPALETTE構造体へのポインタ。

戻り値

正常終了	論理ハーフトーンパレットのハンドル
異常終了	NULL

詳細なエラー情報は、GetLastError関数により取得。

解　説

CreatePalette関数は、論理カラーパレットを作成する。

アプリケーションは、GetDeviceCaps関数を呼び出し、RASTERCAPS定数を指定することによりパレット操作をサポートするデバイスを決めることができる。

アプリケーションは論理パレットを作成すると、SelectPalette関数を呼び出すことによりデバイスコンテキストのパレットを選択することができる。デバイスコンテキストより選択されたパレットは、RealizePalette関数を呼び出すことにより実現化される。

必要なくなったパレットは、DeleteObject関数により削除する。

参　照

DeleteObject,→GetDeviceCaps,→LOGPALETTE,→RealizePalette,→SelectPalette

GetColorAdjustment	gdi32.dll		NT
カラー値の取得			

V C

```
BOOL GetColorAdjustment(
    HDC                 hdc,   //デバイスコンテキストのハンドル
    LPCOLORADJUSTMENT   lpca   //COLORADJUSTMENT構造体へのポインタ
    );
```

V B

```
Declare Function GetColorAdjustment Lib "gdi32" Alias "GetColorAdjustment" (ByVal hdc As Long, lpca As COLORADJUSTMENT) As Long
```

パラメータ

・*hdc*　デバイスコンテキストの識別子。
・*lpca*　色を調整した値を受け取るCOLORADJUSTMENT構造体へのポインタ。

戻り値

正常終了	TRUE
異常終了	FALSE

詳細なエラー情報は、GetLastError関数により取得。

解説

GetColorAdjustment関数は、指定されたデバイスコンテキストに調整されたカラー値を取得する。

参照

→SetColorAdjustment,→COLORADJUSTMENT

GetNearestColor	gdi32.dll	95	NT
表示可能なシステムカラー値取得			

V C

```
COLORREF GetNearestColor(
    HDC         hdc,      //デバイスコンテキストのハンドル
    COLORREF    crColor   //識別させる色
    );
```

V B

Declare Function GetNearestColor Lib "gdi32" Alias "GetNearestColor" (ByVal hdc As Long, ByVal crColor As Long) As Long

パラメータ

・*hdc*　デバイスコンテキストの識別子。
・*crColor*　識別させる色の値を指定する。

戻り値

正常終了	カラー識別子
異常終了	CLR_INVALID

詳細なエラー情報は、GetLastError関数により取得。

解　説

GetNearestColor関数は、指定された色により近い色をシステムパレットから取得する。

参　照

→GetDeviceCaps,→GetNearestPaletteIndex,→COLORREF

GetNearestPaletteIndex	gdi32.dll	95	NT
論理パレットエントリのインデックス取得			

V C

```
UINT GetNearestPaletteIndex(
        HPALETTE     hpal,      //論理カラーパレットのハンドル
        COLORREF     crColor    //識別させる色
        );
```

V B

Declare Function GetNearestPaletteIndex Lib "gdi32" Alias "GetNearestPaletteIndex" (ByVal hPalette As Long, ByVal crColor As Long) As Long

パラメータ

・*hpal*　論理カラーパレットの識別子。
・*crColor*　識別させる色を指定する。

戻り値

正常終了	論理パレットエントリのインデックス
異常終了	CLR_INVALID

詳細なエラー情報は、GetLastError関数により取得。

解説

GetNearestPaletteIndex関数は、指定されたカラー値に一番近い色の論理パレットエントリのインデックスを取得する。
アプリケーションは、GetDeviceCaps関数を呼び出し、RASTERCAPS定数を指定することによりパレット操作をサポートするデバイスを決めることができる。
与えられた論理パレットにPC_EXPLICITフラグが設定されたエントリが含まれた場合の戻り値は、不確定である。

参照

→GetDeviceCaps,→GetNearestColor,→GetPaletteEntries,→GetSystemPaletteEntries,→COLORREF

GetPaletteEntries	gdi32.dll	95	NT
パレットエントリ範囲の取得			

V　C

```
VCUINT GetPaletteEntries(
    HPALETTE        hpal,          //論理カラーパレットのハンドル
    UINT            iStartIndex,   //最初に取得するエントリ
    UINT            nEntries,      //取得するエントリの数
    LPPALETTEENTRY  lppe           //エントリ取得用の配列のアドレス
    );
```

V　B

```
Declare Function GetPaletteEntries Lib "gdi32" Alias "GetPaletteEntries" (ByVal hPalette As Long, ByVal wStartIndex As Long, ByVal wNumEntries As Long, lpPaletteEntries As PALETTEENTRY) As Long
```

パラメータ

- *hpal*　論理カラーパレットの識別子。
- *iStartIndex*　最初に取得する論理パレットのエントリを指定する。
- *nEntries*　取得する論理パレットのエントリ数を指定する。
- *lppe*　パレットエントリを受け取るPALETTEENTRY構造体の配列へのポインタ。この配列は、*nEntries*パラメータで指定された数以上の構造体を保持していなくてはならない。

戻り値

正常終了　論理パレットハンドル≠NULL	取得したエントリ数
正常終了　論理パレットハンドル=NULL	与えられたパレット内のエントリ数
異常終了	0

詳細なエラー情報は、GetLastError関数により取得。

解　説

GetPaletteEntries関数は、与えられた論理パレットから指定されたパレットエントリの範囲を取得する。

アプリケーションは、GetDeviceCaps関数を呼び出し、RASTERCAPS定数を指定することによりパレット操作をサポートするデバイスを決めることができる。

*nEntries*パラメータがパレット内に存在するエントリ以上の数を指定した場合、PALETTEENTRY構造体の残りの要素は変更されない。

参　照

→GetDeviceCaps,→GetSystemPaletteEntries,→PALETTEENTRY

GetSystemPaletteEntries	gdi32.dll	95	NT
システムパレットからパレットエントリ範囲を取得			

V C

```
UINT GetSystemPaletteEntries(
    HDC             hdc,        //デバイスコンテキストのハンドル
    UINT            iStartIndex,//取得する最初のエントリのインデックス
    UINT            nEntries,   //取得するエントリ数
    LPPALETTEENTRY  lppe        //システムパレットエントリを取得する配列
    );
```

V B

Declare Function GetSystemPaletteEntries Lib "gdi32" Alias "GetSystemPaletteEntries"(ByVal hdc As Long, ByVal wStartIndex As Long, ByVal wNumEntries As Long, lpPaletteEntries As PALETTEENTRY) As Long

パラメータ

- *hdc*　デバイスコンテキストの識別子。
- *iStartIndex*　システムパレットから取得された最初のエントリを指定する。
- *nEntries*　システムパレットより取得するエントリ数を指定する。
- *lppe*　パレットエントリを受け取る、PALETTEENTRY構造体の配列へのポインタ。配列は、少なくとも*nEntries*パラメータに指定したエントリ数以上の構造体要素がなければならない。このパラメータがNULLの場合、この関数はパレットのエントリ総数を返す。

戻り値

正常終了	取得したエントリ数
異常終了	0

詳細なエラー情報は、GetLastError関数により取得。

解　説

GetSystemPaletteEntries関数は、指定されたデバイスコンテキストに関連するパレットエントリの範囲をシステムパレットから取得する。
アプリケーションは、GetDeviceCaps関数を呼び出し、RASTERCAPS定数を指定することによりパレット操作をサポートするデバイスを決めることができる。

参　照

→GetDeviceCaps,→GetPaletteEntries,→PALETTEENTRY

GetSystemPaletteUse	gdi32.dll	95	NT
システム（論理）パレットの現在の状態を取得			

V C

```
UINT GetSystemPaletteUse(
    HDC        hdc        //デバイスコンテキストのハンドル
    );
```

V B

Declare Function GetSystemPaletteUse Lib "gdi32" Alias "GetSystemPaletteUse" (ByVal hdc As Long) As Long

パラメータ

・*hdc*　デバイスコンテキストの識別子。

戻り値

関数が正常に終了した場合、システムパレットの現在の状態を返す。以下の値のいずれかになる。

値	意味
SYSPAL_NOSTATIC	システムパレットには、黒と白以外のスタティックカラーがない。
SYSPAL_STATIC	システムパレットには、アプリケーションが論理パレットを実体化したときにも変更されないスタティックカラーが含まれている。
SYSPAL_ERROR	与えられたデバイスコンテキストが不正であるか、カラーパレットをサポートしない。

詳細なエラー情報は、GetLastError関数により取得。

解　説

GetSystemPaletteUse関数は、指定されたデバイスコンテキストのシステム(論理)パレットの現在の状態を取得する。
システムパレットには、20個のデフォルトのスタティックカラーが含まれている。これらの色は、アプリケーションが論理パレットを実体化しても変更されない。アプリケーションはSetSystemPaletteUse関数を呼び出して、これらの色のほとんどにアクセスできる。
*hdc*パレットにより識別するデバイスコンテキストは、カラーパレットをサポートするデバイスを表さなければならない。
アプリケーションは、GetDeviceCaps関数を呼び出し、RASTERCAPS定数を指定することによりパレット操作をサポートするデバイスを決めることができる。

参　照

GetDeviceCaps,SetSystemPaletteUse

RealizePalette	gdi32.dll	95	NT
パレットエントリの割り当て			

V C

```
UINT RealizePalette(
    HDC         hdc     //デバイスコンテキストのハンドル
    );
```

V B

Declare Function RealizePalette Lib "gdi32" Alias "RealizePalette" (ByVal hdc As Long) As Long

パラメータ

・*hdc*　選択された論理パレットを含むデバイスコンテキスト(DC)の識別子。

戻り値

正常終了	システムパレットに割り当てられる論理パレットエントリ数
異常終了	GDI_ERROR

詳細なエラー情報は、GetLastError関数により取得。

解　説

RealizePalette関数は、パレットエントリを、カレント論理パレットからシステムパレットに割り当てる。

アプリケーションは、GetDeviceCaps関数を呼び出し、RASTERCAPS定数を指定することによりパレット操作をサポートするデバイスを決めることができる。

RealizePalette関数は、指定されたデバイスコンテキストに関連するデバイスのパレットを変更する。デバイスコンテキストがメモリであった場合、デバイスコンテキストから選択されたビットマップのカラーテーブルが変更される。デバイスコンテキストがディスプレイであった場合、デバイスの論理パレットが変更される。

論理カラーパレットは、色彩の表現を重視するアプリケーションとシステムとの間で、バッファとして動作する。アプリケーションはこのパレットを使用して、独自に表示する色やほかのウィンドウに表示される色に影響を与えることなく、必要なだけの色数を使用することができる。

アプリケーションのウィンドウがフォーカスを持つときにRealizePalette関数を呼び出すと、Windowsは、要求した可能な限りすべての色を表示できるようにする。同じことが、非アクティブウィンドウについても当てはまる。

参　照

CreatePalette,GetDeviceCaps,SelectPalette

ResizePalette	gdi32.dll	95	NT
論理パレットサイズの拡大縮小			

V　C

```
BOOL ResizePalette(
        HPALETTE     hpal,      //論理パレットのハンドル
        UINT         nEntries   //論理パレットのパレットエントリ数
        );
```

V　B

Declare Function ResizePalette Lib "gdi32" Alias "ResizePalette" (ByVal hPalette As Long, ByVal nNumEntries As Long) As Long

パラメータ

・*hpal*　変更するパレットの識別子。
・*nEntries*　サイズ変更後のパレット内のエントリ数を指定する。

戻り値

正常終了	TRUE
異常終了	FALSE

詳細なエラー情報は、GetLastError関数により取得。

解　説

ResizePalette関数は、指定された値に基づいた論理パレットのサイズを拡大、縮小する。
アプリケーションは、GetDeviceCaps関数を呼び出し、RASTERCAPS定数を指定することによりパレット操作をサポートするデバイスを決めることができる。
アプリケーションでResizePalette関数を呼び出してパレットのサイズを縮小するとき、サイズ変更されたパレット内に残っているエントリは変更されない。ResizePalette関数を呼び出してパレットを拡大するときには、追加されるパレットエントリは黒に設定され(赤、緑、青の値はすべて0)、それらのフラグが0に設定される。

参　照

GetDeviceCaps

SelectPalette	gdi32.dll	95	NT
論理パレットの選択			

V C

```
HPALETTE SelectPalette(
    HDC         hdc,              //デバイスコンテキストのハンドル
    HPALETTE    hpal,             //論理カラーパレットのハンドル
    BOOL        bForceBackground  //前景/背景モード
    );
```

V B

```
Declare Function SelectPalette Lib "gdi32" Alias "SelectPalette" (ByVal hdc As Long,
ByVal hPalette As Long, ByVal bForceBackground As Long) As Long
```

パラメータ

- *hdc*　デバイスコンテキストの識別子。
- *hpal*　選択する論理パレットの識別子。
- *bForceBackground*　論理パレットを常に背景パレットにするかどうかを指定する。この値がTRUEのとき、この関数は可能な限り物理パレット内の色を論理パレットに設定する。実体化されたパレットを持つウィンドウがフォーカスを持たないスレッドに属しているときには、これが実行されている。
この値がFALSEで、アプリケーションが前面にあるとき、RealizePalette関数が論理パレットをデバイスパレットにコピーさせる(*hdc*パラメータがメモリデバイスコンテキストであるとき、このパラメータは無視される)。

戻り値

正常終了	以前の論理パレットの識別子
異常終了	NULL

詳細なエラー情報は、GetLastError関数により取得。

解説

SelectPalette関数は、デバイスコンテキストに、指定された論理パレットを選択する。アプリケーションは、GetDeviceCaps関数を呼び出し、RASTERCAPS定数を指定することによりパレット操作をサポートするデバイスを決めることができる。
アプリケーションは、複数のデバイスコンテキストに同じ論理パレットを選択できる。しかし、論理パレットを変更した場合、すべてのデバイスコンテキストに選択されているパレットに影響を与える。トップレベルのウィンドウの子ウィンドウがそれぞれ自分のパレットを実体化した場合、アプリケーションは、*bForceBackground*パラメータにTRUEを設定してSelectPalette関数を呼び出す。しかし、パレットを実体化する必要のある子ウィンドウだけに、*bForceBackground*パラメータにTRUEを設定しなければならない。他の子ウィンドウには、FALSEを設定する。

参　照

CreatePalette,GetDeviceCaps,RealizePalette

SetColorAdjustment	gdi32.dll	95	NT
デバイスコンテキストへのカラー値設定			

V C

```
BOOL SetColorAdjustment(
    HDC                        hdc,   //デバイスコンテキストのハンドル
    CONST COLORADJUSTMENT* lpca   //COLORADJUSTMENT構造体
                                    へのポインタ
  );
```

V B

Declare Function SetColorAdjustment Lib "gdi32" Alias "SetColorAdjustment" (ByVal hdc As Long, lpca As COLORADJUSTMENT) As Long

パラメータ

・*hdc*　デバイスコンテキストの識別子。
・*lpca*　調整した色の値を格納するCOLORADJUSTMENT構造体へのポインタ。

戻り値

正常終了	TRUE
異常終了	FALSE

詳細なエラー情報は、GetLastError関数により取得。

解　説

SetColorAdjustment関数は、指定された値を使用するデバイスコンテキストに合った色を設定する。
合わせた色は、HALFTONEモードを設定されたときに、StretchBlt関数やStretchDIBits関数を呼び出すためにソースビットマップの入力色を合わせるのに使われる。

参　照

COLORADJUSTMENT,SetStretchBltMode,StretchBlt,StretchDIBits

SetPaletteEntries	gdi32.dll	95	NT
RGB値とフラグの設定			

V C

```
UINT SetPaletteEntries(
    HPALETTE              hpal,      //論理パレットのハンドル
    UINT                  iStart,    //設定する最初のエントリのインデックス
    UINT                  cEntries,  //設定するエントリの数
    CONST PALETTEENTRY*lppe          //構造体の配列のアドレス
    );
```

V B

```
Declare Function SetPaletteEntries Lib "gdi32" Alias "SetPaletteEntries" (ByVal hPalette As Long, ByVal wStartIndex As Long, ByVal wNumEntries As Long, lpPaletteEntries As PALETTEENTRY) As Long
```

パラメータ

- *hpal*　論理パレットの識別子。
- *iStart*　設定する論理パレット内の最初のエントリを指定する。
- *cEntries*　設定する論理パレット内のエントリ数を指定する。
- *lppe*　RGB値とフラグを格納するPALETTEENTRY構造体の配列の最初の要素へのポインタ。

戻り値

正常終了	設定されたエントリ数
異常終了	0

詳細なエラー情報は、GetLastError関数により取得。

解　説

SetPaletteEntries関数は、論理パレットの範囲内のエントリに、RGB(赤、緑、青)カラー値とフラグを設定する。
アプリケーションは、GetDeviceCaps関数を呼び出し、RASTERCAPS定数を指定することによりパレット操作をサポートするデバイスを決めることができる。
論理パレットがデバイスの中に選択され実体化されていると、パレットの変更は表面上、論理パレットに影響しない。新しい論理パレットを設定するためには、RealizePalette関数を呼び出さなければならない。

参　照

GetDeviceCaps,RealizePalette,PALETTEENTRY

SetSystemPaletteUse	gdi32.dll	95	NT
スタティックカラーの使用			

V　C

```
UINT SetSystemPaletteUse(
    HDC     hdc,     //デバイスコンテキストのハンドル
    UINT    uUsage   //パレット使用フラグ
    );
```

V　B

Declare Function SetSystemPaletteUse Lib "gdi32" Alias "SetSystemPaletteUse" (ByVal hdc As Long, ByVal wUsage As Long) As Long

パラメータ

- *hdc*　デバイスコンテキストの識別子。このデバイスコンテキストは、カラーパレットをサポートしなければならない。
- *uUsage*　システムパレットの新しい使い方を指定する。このパラメータは、以下の値のいずれかになる。

値	意味
SYSPAL_NOSTATIC	システムパレットは2種のスタティックカラー(白と黒)を含む。
SYSPAL_STATIC	システムパレットには、論理パレットが実体化されても変更されないスタティックカラーが含まれる。

戻り値

正常終了	システムパレットの以前の設定 (SYSPAL_NOSTATIC,SYSPAL_STATICのいずれか)
異常終了	SYSPAL_ERROR

詳細なエラー情報は、GetLastError関数により取得。

解　説

SetSystemPaletteUse関数は、アプリケーションにシステムパレット内の2または20のスタティックカラーの利用を許可する。デフォルトのシステムパレットは20のスタティックカラーが含まれている(スタティックカラーはアプリケーションが論理パレットを実体化しても変更されない)。
アプリケーションは、GetDeviceCaps関数を呼び出し、RASTERCAPS定数を指定することによりパレット操作をサポートするデバイスを決めることができる。
アプリケーションウィンドウが、前面に移動しSYSPAL_NOSTATIC値が設定されていた場合、アプリケーションは設定されているカレントシステムカラーを保存するためGetSysColor関数を呼び出さなければならない。黒と白のみを使用可能と設定するためには、SetSysColors関数を呼び出す。アプリケーションが背後に戻るか終了したとき、前のシステムカラーを回復しなければならない。

関数がSYSPAL_ERRORを返した場合は、指定されたデバイスコンテキストが不正であるか、カラーパレットがサポートされていない。
アプリケーションは、ウィンドウが最大化されていて入力フォーカスを持っているときだけ、この関数を呼び出さなければならない。
*uUsage*パラメータにSYSPAL_NOSTATICを設定してSetSystemPaletteUse関数を呼び出した場合、Windowsはシステムパレットの2つのエントリのうち白と黒のエントリだけを残す。
*uUsage*パラメータにSYSPAL_NOSTATICを設定してこの関数を呼び出した場合は、アプリケーションは以下の作業を実行しなければならない。

1) 論理パレットを実体化する。
2) GetSysColor関数を呼び出して、カレントシステムカラーの設定を保存する。
3) SetSysColors関数を呼び出し、黒と白を使用してシステムカラーを適切な値に設定する。たとえば、隣り合う項目や重なり合う項目(ウィンドウの枠と境界など)は、それぞれ黒と白とに色分けして設定する。
4) ほかのトップレベルウィンドウにWM_SYSCOLORCHANGEメッセージを送って、それらが新しいシステムカラーを使用して再描画できるようにする。

アプリケーションのウィンドウがフォーカスを失うかクローズした場合には、アプリケーションは以下の作業を実行しなければならない。

1) *uUsage*パラメータにSYSPAL_STATICを設定して、SetSystemPaletteUse関数を呼び出す。
2) 論理パレットを実体化する。
3) システムカラーを元の値に復元する。
4) WM_SYSCOLORCHANGEメッセージを送る。

参　照　　GetDeviceCaps,GetSysColor,SetSysColors

UnrealizeObject	gdi32.dll	95	NT
論理パレットのリセット			

V　C

```
BOOL UnrealizeObject(
    HGDIOBJ    hgdiobj    //論理パレットのハンドル
    );
```

V　B

Declare Function UnrealizeObject Lib "gdi32" Alias "UnrealizeObject" (ByVal hObject As Long) As Long

パラメータ

・*hgdiobj*　リセットする論理パレットの識別子。

戻り値

正常終了	TRUE
異常終了	FALSE

詳細なエラー情報は、GetLastError関数により取得。

解　説

UnrealizeObject関数は、論理パレットをリセットする。この関数は、初期状態で実体化するようにシステムに指示する。その後、アプリケーションが指定されたパレットに対してRealizePalette関数を呼び出すと、システムは論理パレットをシステムパレットに完全に割り当て直す。

*hgdiobj*がブラシであった場合、UnrealizeObject関数は何もせずにTRUEを返す。ブラシの原点を設定するときは、SetBrushOrgEx関数を使う。

UnrealizeObject関数は、ストックオブジェクトに対しては使わない。違うパターンがGetStockObject(DEFAULT_PALETTE)を呼び出し取得されたら、それはストックオブジェクトである。

*hgdiobj*パラメータに識別されるパレットには、デバイスコンテキストに現在選択されているパレットを設定できる。

参　照

GetStockObject,RealizePalette,SetBrushOrgEx

UpdateColors	gdi32.dll	95	NT
クライアント領域の更新			

V C

```
BOOL UpdateColors(
        HDC          hdc      //デバイスコンテキストのハンドル
        );
```

V B

Declare Function UpdateColors Lib "gdi32" Alias "UpdateColors" (ByVal hdc As Long) As Long

パラメータ

・*hdc*　　デバイスコンテキストの識別子。

戻り値

正常終了	TRUE
異常終了	FALSE

詳細なエラー情報は、GetLastError関数により取得。

解　説

UpdateColors関数は、クライアント領域のカレントカラーを実体化された論理パレットに再構成することにより、指定されたデバイスコンテキストのクライアント領域を更新する。
アプリケーションは、GetDeviceCaps関数を呼び出し、RASTERCAPS定数を指定することによりパレット操作をサポートするデバイスを決めることができる。
実体化された論理パレットを持つ非アクティブウィンドウは、システムパレットを変更するときに、クライアント領域を再描画する代わりにUpdateColorsを呼び出して領域を更新できる。
通常は、クライアント領域を再描画するよりもUpdateColors関数を使用した方がすばやく領域を更新する。しかし、UpdateColors関数ではシステムパレットが変更される前の各ピクセルの色に基づいて色を変更するため、色の正確さが損なわれることがある。
この関数は、WM_PALETTECHANGEDメッセージ取得後すぐに呼び出さなければならない。

参　照

GetDeviceCaps,RealizePalette

Coordinate Space and Transformation

ClientToScreen	user32.dll	95	NT
クライアント座標をスクリーン座標に変換			

V　C

```
BOOL ClientToScreen(
        HWND           hWnd,      //変換元座標のウィンドウハンドル
        LPPOINT        lpPoint    //スクリーン座標の構造体へのポインタ
        );
```

V　B

Declare Function ClientToScreen Lib "user32" Alias "ClientToScreen" (ByVal hwnd As Long, lpPoint As POINTAPI) As Long

パラメータ

・*hWnd*　変換されるために使われるクライアント領域を持つウィンドウの識別子。

・*lpPoint*　変換されるクライアント座標を格納している、POINT構造体へのポインタ。新しいスクリーン座標は、この関数が正常に終了したときにこの構造体にコピーされる。

戻り値

正常終了	TRUE
異常終了	FALSE

解　説

ClientToScreen関数は、指定された点のクライアント座標をスクリーン座標に変換する。

ClientToScreen関数は、POINT構造体のクライアント座標をスクリーン座標に置き換える。スクリーン座標は、画面の左上隅からの相対座標である。

参　照

MapWindowPoints,POINT,ScreenToClient

CombineTransform	gdi32.dll		NT
ページ空間への変換			

V C

```
BOOL CombineTransform(
    LPXFORM          lpxformResult,  //変換結果へのポインタ
    CONST XFORM      *lpxform1,      //最初の変換ポインタ
    CONST XFORM      *lpxform2       //2番目の変換ポインタ
    );
```

V B

```
Declare Function CombineTransform Lib "gdi32" Alias "CombineTransform"
(lpxformResult As xform, lpxform1 As xform, lpxform2 As xform) As Long
```

パラメータ

- *lpxformResult*　変換結果を取得するXFORM構造体へのポインタ。
- *lpxform1*　最初の変換を識別するXFORM構造体へのポインタ。
- *lpxform2*　2番目の変換を識別するXFORM構造体へのポインタ。

戻り値

正常終了	TRUE
異常終了	FALSE

解　説

CombineTransform関数は、2WORD空間からページ空間に変換する。最初の変換と2番目の変換は同じ影響を受ける。
3つの変換が必要とされるのは珍しくない。たとえば*lpxform1*は、XFORM構造体を*lpxformResult*と同じように指すことができる。

参　照

GetWorldTransform,ModifyWorldTransform,SetWorldTransform,XFORM

DPtoLP	gdi32.dll	95	NT
デバイス座標を論理座標に変換			

V C

```
BOOL DPtoLP(
    HDC         hdc,        //デバイスコンテキストのハンドル
    LPPOINT     lpPoints,   //座標の配列へのポインタ
    int         nCount      //座標の数
    );
```

V B

```
Declare Function DPtoLP Lib "gdi32" Alias "DPtoLP" (ByVal hdc As Long, lpPoint As POINTAPI, ByVal nCount As Long) As Long
```

パラメータ

- *hdc*　デバイスコンテキストの識別子。
- *lpPoints*　POINT構造体の配列へのポインタ。各POINT構造体のx, y座標が変換される。
- *nCount*　配列中の座標の数を指定する。

戻り値

正常終了	TRUE
異常終了	FALSE

解　説

DPtoLP関数は、デバイス座標を論理座標に変換する。変換後の座標は、デバイスコンテキストのマッピングモード、デバイスのウィンドウとビューポートのそれぞれの原点と大きさの設定そしてワールド変換により決まる。
DPtoLP関数は、デバイス座標が27ビットを超えたときや変換した論理座標が32ビットを超えたとき失敗する。オーバーフローの場合、すべてへのポインタは不確定である。

参　照

LPtoDP,POINT

GetCurrentPositionEx	gdi32.dll	95	NT
デバイスコンテキストのカレント論理座標の取得			

V C

```
BOOL GetCurrentPositionEx(
        HDC          hdc,       //デバイスコンテキストのハンドル
        LPPOINT      lpPoint    //カレント位置を取得する構造体アドレス
        );
```

V B

```
Declare Function GetCurrentPositionEx Lib "gdi32" Alias "GetCurrentPositionEx" (ByVal hdc As Long, lpPoint As POINTAPI) As Long
```

パラメータ

- *hdc*　　デバイスコンテキストの識別子。
- *lpPoint*　　カレント位置の座標を取得するPOINT構造体へのポインタ。

戻り値

正常終了	TRUE
異常終了	FALSE

解説

GetCurrentPositionEx関数は、カレント位置を論理座標で取得する。

参照

MoveToEx,POINT

GetDeviceCaps	gdi32.dll	95	NT
ディスプレイデバイスの固有情報を取得			

V C

```
int GetDeviceCaps(
        HDC          hdc,       //デバイスコンテキストを識別するハンドル
        int          nIndex     //取得する情報のタイプ
        );
```

V B

```
Declare Function GetDeviceCaps Lib "gdi32" Alias "GetDeviceCaps" (ByVal hdc As Long, ByVal nIndex As Long) As Long
```

パラメータ

- *hdc*　　デバイスコンテキストを識別するハンドルを指定する。
- *nIndex*　取得する情報のタイプを指定する。指定する値は次のいずれかとなる。

値	意味
DRIVERVERSION	デバイスドライバのバージョン番号
TECHNOLOGY	デバイス種別。後述の「●TECNOROGYの値」を参照。
HORZSIZE	物理ディスプレイの幅(ミリメートル単位)
VERTSIZE	物理ディスプレイの高さ(ミリメートル単位
HORZRES	ディスプレイの幅(ピクセル単位)
VERTRES	ディスプレイの高さ(ラスター行単位)
LOGPIXELSX	ディスプレイの幅の論理インチ当たりのピクセル数

LOGPIXELSY	ディスプレイの高さの論理インチ当たりのピクセル数
BITSPIXEL	各ピクセルの隣接したカラービットの数
PLANES	カラープレーンの数
NUMBRUSHES	デバイス固有のブラシの数
NUMPENS	デバイス固有のペンの数
NUMMARKERS	デバイス固有のマーカーの数
NUMFONTS	デバイス固有のフォントの数
NUMCOLORS	デバイスのカラーテーブル内のエントリ数
ASPECTX	線の描画に使うデバイスピクセルの相対幅
ASPECTY	線の描画に使うデバイスピクセルの相対高さ
ASPECTXY	線の描画に使うデバイスピクセルの対角線の幅
PDEVICESIZE	PDEVICE 内部構造体のサイズ(バイト単位)
CLIPCAPS	デバイスがサポートするクリッピング能力。後述の「●CLIPCAPSの値」を参照。
RASTERCAPS	デバイスがサポートするラスター能力。後述の「●RASTERSCAPSの値」を参照。
RC_GDI20_STATE	デバイスコンテキスト内に状態ブロックを含む
RC_NONE	ラスター演算をサポートしない
RC_OP_DX_OUTPUT	非透過デバイスとDX配列をサポート
RC_PALETTE	パレットベースのデバイスを指定
RC_SAVEBITMAP	ビットマップをローカルに保存
RC_SCALING	スケーリングをサポート
RC_STRETCHBLT	StretchBlt関数をサポート
RC_STRETCHDIB	StretchDIBits関数をサポート
CURVECAPS	デバイスがサポートする曲線描画能力。後述の「●CURVECAPSの値」を参照。
LINECAPS	デバイスがサポートする線分描画能力。後述の「●LINECAPSの値」を参照。
POLYGONALCAPS	デバイスがサポートする多角形描画能力。後述の「●POLYGONALCAPSの値」を参照。
TEXTCAPS	デバイスがサポートするテキストの表示能力。後述の「●TEXTCAPSの値」を参照。

●TECHNOLOGYの値

TECHNOLOGY（デバイス種別）は、次の値のいずれか。

値	意味
DT_PLOTTER	ベクタープロッタ
DT_RASDISPLAY	ラスターディスプレイ
DT_RASPRINTER	ラスタープリンタ
DT_RASCAMERA	ラスターカメラ
DT_CHARSTREAM	文字ストリーム
DT_METAFILE	メタファイル
DT_DISPFILE	ディスプレイファイル

●CLIPCAPSの値

CLIPCAPS（デバイスがサポートするクリッピング能力）は、次の値のいずれか。

値	意味
CP_NONE	出力はクリップされません。
CP_RECTANGLE	出力は長方形にクリップされます。
CP_REGION	出力はリージョンにクリップされます。
SIZEPALETTE	システムパレット内のエントリ数。このインデックスは、デバイスドライバがRASTERCAPSインデックスにRC_PALETTEビットをセットしているときだけ有効で、Windows3.0以降に対応するドライバだけが使用できる。
NUMRESERVED	システムパレット内の予約エントリ数。このインデックスは、デバイスドライバがRASTERCAPSインデックスにRC_PALETTEビットをセットしているときだけ有効で、Windows3.0以降に対応するドライバだけが使用できる。
COLORRES	デバイスのカラー解像度(ピクセル当たりのビット数)。デバイスドライバがRASTERCAPSインデックスにRC_PALETTEビットをセットしているときだけ有効で、Windows3.0以降に対応するドライバだけが使用できる。

●RASTERCAPSの値

RASTERCAPS（デバイスがサポートするラスター能力）は、次の値の組み合わせ。

値	意味
RC_BANDING	バンド処理をサポート
RC_BIGFONT	64KBより大きいフォントをサポート
RC_BITBLT	ビットマップを転送
RC_BITMAP64	64KBより大きいビットマップをサポート
RC_DEVBITS	デバイスビットマップをサポート
RC_DI_BITMAP	SetDIBits関数とGetDIBits 関数をサポート
RC_DIBTODEV	SetDIBitsToDevice関数をサポート
RC_FLOODFILL	塗りつぶしを実行
RC_GDI20_OUTPUT	Windows 2.0機能をサポート

●CURVECAPSの値

CURVECAPS（デバイスがサポートする曲線描画能力）は、次の値の組み合わせ。

値	意味
CC_NONE	曲線をサポートしない
CC_CIRCLES	円をサポート
CC_PIE	扇形をサポート
CC_CHORD	弓形をサポート
CC_ELLIPSES	楕円をサポート
CC_WIDE	幅のある境界をサポート
CC_STYLED	スタイルを持つ境界をサポート
CC_WIDESTYLED	スタイルを持ち、幅のある境界をサポート
CC_INTERIORS	内部の塗りつぶしをサポート
CC_ROUNDRECT	丸い角を持つ長方形をサポート

●LINECAPSの値

LINECAPS（デバイスがサポートする線分描画能力）は、次の値の組み合わせ。

値	意味
LC_NONE	線分をサポートしない
LC_POLYLINE	折れ線をサポート
LC_MARKER	マーカーをサポート
LC_POLYMARKER	ポリマーカーをサポート
LC_WIDE	幅のある線分をサポート
LC_STYLED	スタイルを持つ線分をサポート
LC_WIDESTYLED	スタイルを持ち、幅のある線分をサポート
LC_INTERIORS	内部の塗りつぶしをサポート

●POLYGONALCAPSの値

POLYGONALCAPS（デバイスがサポートする多角形描画能力）は、次の値の組み合わせ。

値	意味
PC_NONE	多角形をサポートしない
PC_POLYGON	交互モードでの多角形の塗りつぶしをサポート
PC_RECTANGLE	長方形をサポート
PC_WINDPOLYGON	全域モードでの多角形の塗りつぶしをサポート
PC_SCANLINE	走査線の描画をサポート
PC_WIDE	幅のある境界をサポート
PC_STYLED	スタイルを持つ境界をサポート
PC_WIDESTYLED	スタイルを持ち，幅のある境界の描画をサポート
PC_INTERIORS	内部の塗り

●TEXTCAPSの値

TEXTCAPS（デバイスがサポートするテキストの表示能力）は、次の値の組み合わせ。

値	意味
TC_OP_CHARACTER	文字出力精度をサポート。デバイスがデバイスフォントを任意のピクセル位置に置けることを示す。この値は、デバイスフォントを持つすべてのデバイスに対して必要。
TC_OP_STROKE	ストローク出力精度をサポート。デバイスがデバイスフォントの任意のストロークを省略できることを示す。
TC_CP_STROKE	ストロークククリップ精度をサポートする。デバイスがデバイスフォントをピクセル境界にクリップできることを示す。
TC_CR_90	90度の文字の回転をサポート。デバイスが文字を一度に90度だけ回転できることを示す。
TC_CR_ANY	任意の角度の文字の回転をサポート。デバイスが任意の角度でデバイスフォントを回転できることを示す。
TC_SF_X_YINDEP	x方向とy方向の両方のスケーリングをサポート。デバイスがx方向とy方向に分けてデバイスフォントをスケーリングできることを示す。
TC_SA_DOUBLE	倍角文字のスケーリングをサポート。デバイスがデバイスフォントのサイズを2倍にできることを示す。
TC_SA_INTEGER	整数倍のスケーリングをサポート。デバイスが任意の整数倍にデバイスフォントのサイズをスケーリングできることを示す。
TC_SA_CONTIN	任意の整数倍の正確なスケーリングをサポート。デバイスがデバイスフォントを任意の値でスケーリングでき、その上でxとyの比率を保つことを示す。
TC_EA_DOUBLE	太字体の文字をサポート。デバイスがデバイスフォントに太字体を使えることを示す。プリンタドライバでこのビットがセットされないとき、グラフィックデバイスインターフェイス(GDI)は文字を2回印刷して太字体のデバイスフォントを作成する。
TC_IA_ABLE	斜体をサポート。デバイスがデバイスフォントに斜体を使えることを示す。このビットがセットされないとき、GDIは斜体を使用できないものと仮定する。

TC_UA_ABLE	下線をサポート。デバイスがデバイスフォントに下線を付けられることを示す。このビットがセットされないとき、GDIはデバイスフォント用の下線を作成する。
TC_SO_ABLE	打ち消し線をサポート。デバイスがデバイスフォントに打ち消し線を付けられることを示す。このビットがセットされないとき、GDIはデバイスフォント用の打ち消し線を作成する。
TC_RA_ABLE	ラスターフォントをサポート。GDIがEnumFonts関数またはEnumFontFamilies関数の呼び出しに応答して、このデバイスで利用できる任意のラスターフォントやTrueTypeフォントを列挙することを示す。このビットがセットされないときは、これらの関数が呼び出されたときに、GDIが供給するラスターフォントやTrueTypeフォントが列挙されない。
TC_VA_ABLE	ベクターフォントをサポート。GDIがEnumFonts関数またはEnumFontFamilies 関数の呼び出しに応答して、このデバイスで利用できる任意のベクターフォントやTrueTypeフォントを列挙することを示す。これはベクターデバイス(すなわちプロッタ)に対してだけ有効。GDIはベクターフォントをドライバに送る前にラスタライズするので、ディスプレイドライバ(ラスターフォントの表示能力を持つもの)とラスタープリンタドライバは、常にベクターフォントを列挙する。
TC_RESERVED	予約。0でなければならない。

戻り値

正常終了	要求された値を返す

参照

CreateEnhMetaFile,CreateIC,DeviceCapabilities,GetDIBits,GetObjectType, SetDIBits,SetDIBitsToDevice,StretchBlt,StretchDIBits

GetGraphicsMode	gdir32.dll	95	NT

デバイスコンテキストのカレントグラフィックモードの回復

VC

```
int GetGraphicsMode(
        HDC          hdc       //デバイスコンテキストのハンドル
        );
```

VB

Declare Function GetGraphicsMode Lib "gdi32" Alias "GetGraphicsMode" (ByVal hdc As Long) As Long

パラメータ

・*hdc*　　デバイスコンテキストの識別子。

戻り値

関数が正常に終了した場合、グラフィックモードとして以下の値が設定される。

値	意味
GM_COMPATIBLE	カレントグラフィックモードは、Windowsバージョン3.1と互換性がある。このグラフィックモード内でアプリケーションは、指定されたデバイスコンテキストのワールド変換に設定することや更新ができない。互換性のあるグラフィックモードが、デフォルトのグラフィックモードである。
GM_ADVANCED	WindowsNTでは、カレントグラフィックモードは向上したグラフィックモードであり、ワールド変換を許したモードである。このグラフィックモード内でアプリケーションは、指定されたデバイスコンテキストのワールド変換を設定したり更新することができる。Windows 95では、GM_ADVANCEDはサポートされていない。
それ以外	0

解説

GetGraphicsMode関数は、指定されたデバイスコンテキストをカレントのグラフィックモードに回復する。
アプリケーションは、SetGraphicsMode関数を呼び出すことによりデバイスコンテキストのグラフィックモードを設定できる。

参照

SetGraphicsMode

GetMapMode	gdi32.dll	95	NT
カレントマッピングの取得			

V C

```
int GetMapMode(
    HDC        hdc      //デバイスコンテキストのハンドル
    );
```

V B

```
Declare Function GetMapMode Lib "gdi32" Alias "GetMapMode" (ByVal hdc As Long) As Long
```

パラメータ

・*hdc*　　デバイスコンテキストの識別子。

戻り値

正常終了	マッピングモードを示す値
異常終了	0

解　説

GetMapMode関数は、カレントマッピングモードを取得する。
マッピングモードは、以下の値のいずれかである。

値	意味
MM_ANISOTROPIC	論理単位は、任意にスケーリングされた軸上の任意の単位に変換される。単位、向き、スケーリングを指定するときには、SetWindowExtEx関数とSetViewportExtEx関数を使う。
MM_HIENGLISH	各論理単位は、0.001インチに変換される。x座標の正方向は右、y座標の正方向は上になる。
MM_HIMETRIC	各論理単位は、0.01ミリメートルに変換される。x座標の正方向は右、y座標の正方向は上になる。
MM_ISOTROPIC	各論理単位は、等しくスケーリングされた軸上の任意の単位に変換される。x軸方向の1単位は、y軸方向の1単位と同じ。両軸の単位と向きを指定するときには、SetWindowExtEx関数とSetViewportExtEx関数を使う。グラフィックデバイスインターフェイス(GDI)は、必要に応じて、x単位とy単位のサイズが同一になるように調整する(たとえば、ウィンドウ範囲を設定するとき、ビューポートが常にその単位が同じであるよう合わせる)。
MM_LOENGLISH	各論理単位は、0.01インチに変換される。x座標の正方向は右に、y座標の正方向は上になる。
MM_LOMETRIC	各論理単位は、0.1ミリメートルに変換される。x座標の正方向は右に、y座標の正方向は上になる。
MM_TEXT	各論理単位は、1デバイスピクセルに変換される。x座標の正方向は右に、y座標の正方向は下になる。
MM_TWIPS	各論理単位は、ポイント数の20分の1(1440分の1インチ)に変換される。x座標の正方向は右に、y座標の正方向は上になる。

参　照

SetMapMode,SetWindowExtEx,SetViewportExtEx

GetViewportExtEx	gdi32.dll	95	NT
デバイスコンテキストのカレントビューポート範囲の取得			

V C

```
BOOL GetViewportExtEx(
        HDC        hdc,        //デバイスコンテキストのハンドル
        LPSIZE     lpSize      //ビューポート範囲を取得する構造体へのポインタ
        );
```

V B

Declare Function GetViewportExtEx Lib "gdi32" Alias "GetViewportExtEx"(ByVal hdc As Long, lpSize As SIZE) As Long

パラメータ

- *hdc*　デバイスコンテキストの識別子。
- *lpSize*　SIZE構造体へのポインタ。デバイス単位のx範囲とy範囲がこの構造体に格納される。

戻り値

正常終了	TRUE
異常終了	FALSE

解説

GetViewportExtEx関数は、指定されたデバイスコンテキストのカレントビューポートのx範囲とy範囲を取得する。

参照

GetWindowExtEx,SetViewportExtEx,SetWindowExtEx

GetWindowExtEx	gdi32.dll	95	NT
デバイスコンテキストに対応するウィンドウの範囲を取得			

V C

```
BOOL GetWindowExtEx(
    HDC       hdc,     //デバイスコンテキストのハンドル
    LPSIZE    lpSize   //ウィンドウ範囲を取得する構造体アドレス
    );
```

V B

```
Declare Function GetWindowExtEx Lib "gdi32" Alias "GetWindowExtEx" (ByVal hdc
As Long, lpSize As SIZE) As Long
```

パラメータ

- *hdc*　デバイスコンテキストの識別子。
- *lpSize*　SIZE構造体へのポインタ。ページ単位のx範囲とy範囲がこの構造体に格納される。

戻り値

正常終了	TRUE
異常終了	FALSE

解　説

GetWindowExtEx関数は、指定されたデバイスコンテキストに対応するウィンドウのx範囲とy範囲を取得する。

参　照

GetViewportExtEx,SetViewportExtEx,SetWindowExtEx

GetWorldTransform	gdi32.dll	95	NT
ワールド空間からページ空間への変換の取得			

V　C

```
BOOL GetWorldTransform(
    HDC         hdc,       //デバイスコンテキストのハンドル
    LPXFORM     lpXform    //変換結果を取得する構造体へのポインタ
    );
```

V　B

Declare Function GetWorldTransform Lib "gdi32" Alias "GetWorldTransform" (ByVal hdc As Long, lpXform As xform) As Long

パラメータ

- *hdc*　　デバイスコンテキストのハンドル。
- *lpXform*　　カレントのワールド空間よりページ空間へ変換した結果を取得するXFORM構造体へのポインタ。

戻り値

正常終了	TRUE
異常終了	FALSE

解　説

GetWorldTransform関数は、ワールド空間よりページ空間に変換した結果を取得する。

アプリケーションがGetWorldTransform関数を呼び出す前にModifyWorldTransform関数を呼び出すと、変換の精度は変更される(FLOAT値より高い精度を使う変換値を保存するため内部フォーマットを作るため)。

参　照

ModifyWorldTransform,SetWorldTransform

LPtoDP	gdi32.dll	95	NT
論理座標をデバイス座標に変換			

V C

```
BOOL LPtoDP(
    HDC          hdc,        //デバイスコンテキストのハンドル
    LPPOINT      lpPoints,   //点の配列へのポインタ
    int          nCount      //点の数
    );
```

V B

```
Declare Function LPtoDP Lib "gdi32" Alias "LPtoDP" (ByVal hdc As Long, lpPoint As POINTAPI, ByVal nCount As Long) As Long
```

パラメータ

- *hdc*　デバイスコンテキストの識別子。
- *lpPoints*　POINT構造体の配列へのポインタ。各POINT構造体のx座標とy座標は、変換される。
- *nCount*　配列の点の数を指定する。

戻り値

正常終了	TRUE
異常終了	FALSE

解　説

LPtoDP関数は、論理座標をデバイス座標に変換する。変換はデバイスコンテキストのマッピングモードや、ウィンドウとビューポートの原点と範囲そしてワールド変換の設定により異なる。

論理座標が32ビットを超えたときや変換したデバイス座標が27ビットを超えたとき、この関数は失敗する。オーバーフローの場合、すべてへのポインタは不確定である。

参　照

DPtoLP,POINT

MapWindowPoints	user32.dll	95	NT
点の集合を指定ウィンドウの座標空間へ変換			

V C

```
int MapWindowPoints(
        HWND      hWndFrom,   //マップ元のウィンドウのハンドル
        HWND      hWndTo,     //マップ先のウィンドウのハンドル
        LPPOINT   lpPoints,   //マップするPOINT構造体へのポインタ
        UINT      cPoints     //配列中の構造体の個数
        );
```

V B

Declare Function MapWindowPoints Lib "user32" Alias "MapWindowPoints" (ByVal hwndFrom As Long, ByVal hwndTo As Long, lppt As Any, ByVal cPoints As Long) As Long

パラメータ

- *hWndFrom*　点の変換元のウィンドウの識別子。このパラメータがNULLまたはHWND_DESKTOPのときは、点はスクリーン座標であると仮定される。
- *hWndTo*　点の変換先のウィンドウの識別子。このパラメータがNULLまたはHWND_DESKTOPのときは、点はスクリーン座標であると仮定される。
- *lpPoints*　変換する座標を含むPOINT構造体の配列へのポインタ。このパラメータには、RECT構造体へのポインタを指定することもできる。この場合、*cPoints*パラメータに2を設定する。
- *cPoints*　*lpPoints*パラメータが指す配列のPOINT構造体の数を指定する。

戻り値

関数が正常に終了した場合、戻り値の下位ワードには、それぞれの目標点の水平座標を算出するため、基点の水平座標に加算されたピクセル数が設定される。上位ワードには、それぞれの目標点の垂直座標を算出するため、基点の垂直座標に加算されたピクセル数が設定される。

解説

MapWindowPoints関数は、点の集合を、ウィンドウに相対な座標空間からほかのウィンドウに相対な座標空間へ変換(マップ)する。

参照

ClientToScreen,POINT,RECT,ScreenToClient

ModifyWorldTransform	gdi32.dll		NT
デバイスコンテキストのワールド変換			

V C

```
BOOL ModifyWorldTransform(
    HDC             hdc,        //デバイスコンテキストのハンドル
    CONST XFORM*    lpXform,    //変換データのアドレス
    DWORD           iMode       //変換モード
    );
```

V B

```
Declare Function ModifyWorldTransform Lib "gdi32" Alias "ModifyWorldTransform"
(ByVal hdc As Long, lpXform As xform, ByVal iMode As Long) As Long
```

パラメータ

- *hdc*　デバイスコンテキストの識別子。
- *lpXform*　与えられたデバイスコンテキストのワールド変換を更新するため使用されるXFORM構造体へのポインタ。
- *iMode*　変換データをカレントのワールド変換に更新する方法を指定する。このパラメータには、以下の値が設定される。

値	意味
MWT_IDENTITY	同一マトリックスを使うことによりカレントのワールド変換を再設定する。このモードが指定されたときは、*lpXform*により指されたXFORM構造体は無視される。
MWT_LEFTMULTIPLY	XFORM構造体のデータによりカレントの変換を増やす(XFORM構造体のデータは左の被乗数になる。カレントの変換のデータは右の被乗数になる)。
MWT_RIGHTMULTIPLY	XFORM構造体のデータによりカレントの変換を増やす(XFORM構造体のデータは右の被乗数になる。カレントの変換のデータは左の被乗数になる)。

戻り値

正常終了	TRUE
異常終了	FALSE

解説

ModifyWorldTransform関数は、指定されたモードでデバイスコンテキストのワールド変換を変える。

ModifyWorldTransform関数は、あらかじめSetGraphicsMode関数を呼び出しGM_ADVANCEDが設定されている指定されたデバイスコンテキストのグラフィックモードでなければ失敗する。なおかつ、ワールド変換が最初にSetWorldTransform関数やModifyWorldTransform関数を呼び出し、デフォルトの内部変換に再設定しなければ、デフォルトのGM_COMPATIBLEモードにデバイスコンテキストのグラフィックモードを再設定することはできない。

参照

GetWorldTransform,SetWorldTransform,SetGraphicsMode,XFORM

OffsetViewportOrgEx	gdi32.dll	95	NT
デバイスコンテキストのビューポートをオフセット移動			

V　C

```
BOOL OffsetViewportOrgEx(
    HDC       hdc,       //ドライブコンテキストのハンドル
    int       nXOffset,  //水平方向のオフセット
    int       nYOffset,  //垂直方向のオフセット
    LPPOINT   lpPoint    //オリジナルの原点を取得する構造体へのポインタ
    );
```

V　B

Declare Function OffsetViewportOrgEx Lib "gdi32" Alias "OffsetViewportOrgEx" (ByVal hdc As Long, ByVal nX As Long, ByVal nY As Long, lpPoint As POINTAPI) As Long

パラメータ

- *hdc*　デバイスコンテキストの識別子。
- *nXOffset*　水平方向のオフセットをデバイス単位で指定する。
- *nYOffset*　垂直方向のオフセットをデバイス単位で指定する。
- *lpPoint*　POINT構造体へのポインタ。以前のビューポートの原点の座標がデバイス単位でこの構造体に返される。*lpPoint*がNULLのときは、以前のビューポートの原点は返されない。

戻り値

正常終了	TRUE
異常終了	FALSE

解　説

OffsetViewportOrgEx関数は、デバイスコンテキストでのビューポートの原点の座標を指定された水平・垂直方向オフセットにより修正する。
新しい原点は、直前の原点に水平・垂直方向オフセットを加算した値になる。

参　照

GetViewportOrgEx,OffsetWindowOrgEx,SetViewportOrgEx

OffsetWindowOrgEx	gdi32.dll	95	NT
デバイスコンテキストの原点をオフセット移動			

V. C

```
BOOL OffsetWindowOrgEx(
    HDC       hdc,       //デバイスコンテキストのハンドル
    int       nXOffset,  //水平方向のオフセット
    int       nYOffset,  //垂直方向のオフセット
    LPPOINT   lpPoint    //オリジナルの原点を取得する構造体へのポインタ
    );
```

V B

```
Declare Function OffsetWindowOrgEx Lib "gdi32" Alias "OffsetWindowOrgEx"
(ByVal hdc As Long, ByVal nX As Long, ByVal nY As Long, lpPoint As POINTAPI)
As Long
```

パラメータ

- *hdc*　デバイスコンテキストの識別子。
- *nXOffset*　水平方向のオフセットを論理単位で指定する。
- *nYOffset*　垂直方向のオフセットを論理単位で指定する。
- *lpPoint*　POINT構造体へのポインタ。以前のウィンドウの原点の論理座標でこの構造体に返される。*lpPoint*がNULLのときは、以前のビューポートの原点は返されない。

戻り値

正常終了	TRUE
異常終了	FALSE

解　説

OffsetWindowOrgEx関数は、デバイスコンテキストでのウィンドウの原点を、指定された水平・垂直方向オフセットにより修正する。

参　照

GetViewportOrgEx,OffsetViewportOrgEx,POINT

ScaleViewportExtEx	gdi32.dll	95	NT
デバイスコンテキストのビューポートを指定率でリサイズ			

V　C

```
BOOL ScaleViewportExtEx(
    HDC      hdc,     //デバイスコンテキストのハンドル
    int      Xnum,    //水平方向に掛ける値
    int      Xdenom,  //水平方向を割る値
    int      Ynum,    //垂直方向に掛ける値
    int      Ydenom,  //垂直方向を割る値
    LPSIZE   lpSize   //以前のビューポート範囲へのポインタ
    );
```

V　B

```
Declare Function ScaleViewportExtEx Lib "gdi32" Alias "ScaleViewportExtEx" (ByVal hdc As Long, ByVal nXnum As Long, ByVal nXdenom As Long, ByVal nYnum As Long, ByVal nYdenom As Long, lpSize As SIZE) As Long
```

パラメータ

- *hdc*　デバイスコンテキストの識別子。
- *Xnum*　現在の水平方向に掛ける値を指定する。
- *Xdenom*　現在の水平方向を割る値を指定する。
- *Ynum*　現在の垂直方向に掛ける値を指定する。
- *Ydenom*　現在の垂直方向を割る値を指定する。
- *lpSize*　SIZE構造体へのポインタ。以前のビューポート範囲(デバイス単位)は、この構造体に格納される。*lpSize*がNULLのときは、何も返されない。

戻り値

正常終了	TRUE
異常終了	FALSE

解　説

ScaleViewportExtEx関数は、デバイスコンテキストのビューポートの範囲を指定された掛ける値や割る値で形成された割合により修正する。
ビューポート範囲は、以下のように修正される。

xNewVE=(xOldVE**Xnum*) / *Xdenom*
yNewVE=(yOldVE**Ynum*) / *Ydenom*

参　照

GetViewportExtEx,SIZE

ScaleWindowExtEx	gdi32.dll	95	NT
デバイスコンテキストのウィンドウを指定率でリサイズ			

V C

```
BOOL ScaleWindowExtEx(
    HDC       hdc,      //デバイスコンテキストのハンドル
    int       Xnum,     //水平方向に掛ける値
    int       Xdenom,   //水平方向を割る値
    int       Ynum,     //垂直方向に掛ける値
    int       Ydenom,   //垂直方向を割る値
    LPSIZE    lpSize    //以前のウィンドウ範囲のアドレス
    );
```

V B

Declare Function ScaleWindowExtEx Lib "gdi32" Alias "ScaleWindowExtEx" (ByVal hdc As Long, ByVal nXnum As Long, ByVal nXdenom As Long, ByVal nYnum As Long, ByVal nYdenom As Long, lpSize As SIZE) As Long

パラメータ

- *hdc*　デバイスコンテキストの識別子。
- *Xnum*　現在の水平方向に掛ける値を指定する。
- *Xdenom*　現在の水平方向を割る値を指定する。
- *Ynum*　現在の垂直方向に掛ける値を指定する。
- *Ydenom*　現在の垂直方向を割る値を指定する。
- *lpSize*　SIZE構造体へのポインタ。以前のウィンドウ範囲(論理単位)は、この構造体に格納される。*lpSize*がNULLのときは、何も返されない。

戻り値

正常終了	TRUE
異常終了	FALSE

解説

ScaleWindowExtEx関数は、デバイスコンテキストのウィンドウを指定された掛ける値や割る値で形成された割合により修正する。
ウィンドウ範囲は、以下のように修正される。

xNewWE=(xOldWE**Xnum*) / *Xdenom*
yNewWE=(yOldWE**Ynum*) / *Ydenom*

参照

GetWindowExtEx,SIZE

ScreenToClient	user32.dll	95	NT
スクリーン座標をクライアント座標に変換			

V C

```
BOOL ScreenToClient(
    HWND        hWnd,     //ソース座標のウィンドウのハンドル
    LPPOINT     lpPoint   //座標の構造体へのポインタ
    );
```

V B

```
Declare Function ScreenToClient Lib "user32" Alias "ScreenToClient" (ByVal hwnd As Long, lpPoint As POINTAPI) As Long
```

パラメータ

- *hWnd*　座標が変換されるクライアント領域を持つウィンドウの識別子。
- *lpPoint*　変換されるスクリーン座標を格納している、POINT構造体へのポインタ。

戻り値

正常終了	TRUE
異常終了	FALSE

解　説

ScreenToClient関数は、画面上の指定された点のスクリーン座標をクライアント座標に変換する。
この関数は、*hWnd*パラメータにより識別されるウィンドウを使い、POINT構造体のスクリーン座標をクライアント座標に算出する。新しい座標は、指定されたウィンドウのクライアント領域の左上隅からの相対座標になる。
ScreenToClient関数は、指定された点をスクリーン座標として仮定する。

参　照

ClientToScreen,MapWindowPoints,POINT

SetGraphicsMode	gdi32.dll	95	NT
デバイスコンテキストのグラフィックモード設定			

V C

```
int SetGraphicsMode(
    HDC         hdc,      //デバイスコンテキストのハンドル
    int         iMode     //グラフィックモード
    );
```

V B

```
Declare Function SetGraphicsMode Lib "gdi32" Alias "SetGraphicsMode" (ByVal hdc As Long, ByVal iMode As Long) As Long
```

パラメータ

- *hdc*　デバイスコンテキストのハンドル。
- *iMode*　グラフィックモードを指定する。このパラメータには以下の値が設定される。

値	意味
GM_COMPATIBLE	カレントグラフィックモードは、Windows3.1と互換性がある。これがデフォルトモードである。アプリケーションは、SetViewportExtEx関数やSetWindowExtEx関数を呼び出すことで、ワールド変換からデバイス変換に更新できる。
GM_ADVANCED	WindowsNTでは、ワールド変換を許したグラフィックモードを設定する。アプリケーションが指定されたデバイスコンテキストのワールド変換を設定したり更新するときは、この値を設定する。すべてのグラフィックで、テキスト出力を含むこのモードは、指定されたデバイスコンテキストでワールドからデバイス変換を完全に行う。Windows 95では、GM_ADVANCEDはサポートされていない。しかし、エンハンストメタファイルはGM_ADVANCEDモードで実行されると仮定される。Windows 95は、エンハンストメタファイルをWindowsNTと同じように見せることを保証する。

戻り値

正常終了	以前のグラフィックモード
異常終了	0

解　説

SetGraphicsMode関数は、指定されたデバイスコンテキストのグラフィックモードを設定する。
グラフィックモードにより異なるグラフィック出力には3つの領域がある。

1. テキスト出力。GM_COMPATIBLEモードで、TrueType(またはベクトルフォント)のテキスト出力は、デバイスコンテキストのワールドからデバイスへの変換に合わせるラスタフォントのテキスト出力と同様の動作をする。グラフィックの静止位置がx軸やy軸に転回されるとき、TrueTypeのテキストは、左側から右側へ上から書かれる。TrueType(またはベクトルフォント)のテキストは、適当な高さに決まる。GM_COMPATIBLEモードで水平でないテキストを書く方法は、デバイスコンテキストで選択された論理フォントに0でない傾斜と方向を指定する。GM_ADVANCEDモードでは、TrueType(またはベクトルフォント)のテキスト出力は、デバイスコンテキストのワールドからデバイスへの変換に合わせる。ラスターフォントが限定された変換能力を持つ。グラフィックデバイスインターフェイス(GDI)は、特殊な変換のためラスターフォントを持ち、最善の出力を行おうとする。

2. 矩形の排除。デフォルトのGM_COMPATIBLEグラフィックモードが設定されたときは、システムは矩形を描くときに、下部と最右端を除外する。アプリケーションが右下を含めた矩形を描くときは、GM_ADVANCEDグラフィックモードが要求される。

3. 弧の描画。デフォルトのGM_COMPATIBLEグラフィックモードが設定されたときは、GDIはデバイス空間にカレントの弧の方向に向けて弧を描く。この約束の中で、弧はx軸やy軸に沿った転回を要求するページからデバイスへの変換を優先しない。

GM_ADVANCEDグラフィックモードが設定されたとき、GDIは論理空間に左回りの方向に弧を描く。これは、GM_ADVANCEDグラフィックモードで弧の制御点と弧自体がデバイスコンテキストのワールドからデバイスへの変換を優先するということである。

参　照　CreateDC,GetArcDirection,GetDC,GetGraphicsMode,ModifyWorldTransform,SetArcDirection,SetViewportExtent,SetViewportExtEx,SetWindowExtent,SetWindowExtEx,SetWorldTransform

SetMapMode	gdi32.dll	95	NT
デバイスコンテキストのマッピングモード設定			

V　C

```
int SetMapMode(
    HDC     hdc,          //デバイスコンテキストのハンドル
    int     fnMapMode     //新しいマッピングモード
    );
```

V　B

```
Declare Function SetMapMode Lib "gdi32" Alias "SetMapMode" (ByVal hdc As Long, ByVal nMapMode As Long) As Long
```

パラメータ

- *hdc*　デバイスコンテキストの識別子。
- *fnMapMode*　新しいマッピングモードを指定する。このパラメータは、以下の値のいずれかになる。

値	意味
MM_ANISOTROPIC	論理単位は、任意にスケーリングされた軸上の任意の単位に変換される。単位、向き、スケーリングを指定するときには、SetWindowExtEx関数とSetViewportExtEx関数を使う。
MM_HIENGLISH	各論理単位は、0.001インチに変換される。x座標の正方向は右、y座標の正方向は上になる。
MM_HIMETRIC	各論理単位は、0.01ミリメートルに変換される。x座標の正方向は右、y座標の正方向は上になる。
MM_ISOTROPIC	各論理単位は、等しくスケーリングされた軸上の任意の単位に変換される。x軸方向の1単位は、y軸方向の1単位と同じ。両軸の単位と向きを指定するときには、SetWindowExtEx関数とSetViewportExtEx関数を使う。グラフィックデバイスインターフェイス(GDI)は、必要に応じて、x単位とy単位のサイズが同一になるように調整する(たとえば、ウィンドウ範囲を設定するとき、ビューポートが常にその単位が同じであるよう合わせる)。
MM_LOENGLISH	各論理単位は、0.01インチに変換される。x座標の正方向は右に、y座標の正方向は上になる。
MM_LOMETRIC	各論理単位は、0.1ミリメートルに変換される。x座標の正方向は右に、y座標の正方向は上になる。
MM_TEXT	各論理単位は、1デバイスピクセルに変換される。x座標の正方

	向は右に、y座標の正方向は下になる。
MM_TWIPS	各論理単位は、ポイント数の20分の1(1440分の1インチ)に変換される。x座標の正方向は右に、y座標の正方向は上になる。

戻り値

正常終了	以前のマッピングモード
異常終了	0

解　説

SetMapMode関数は、指定されたデバイスコンテキストのマッピングモードを設定する。マッピングモードとは、ページ単位をデバイス単位に変換するときの比率を定義し、デバイスのx軸とy軸の向きも合わせて定義する。

MM_TEXTモードを使用すると、アプリケーションは1ピクセルを1論理単位とするデバイスピクセル単位で作業できる。

MM_HIENGLISH、MM_HIMETRIC、MM_LOENGLISH、MM_LOMETRIC、MM_TWIPSのモードは、アプリケーションが物理単位(インチやミリメートルなど)で描画するときに便利である。

MM_ISOTROPICモードは縦横比が1:1である。MM_ANISOTROPICモードを使用すると、x座標とy座標を調節できる。

参　照

GetMapMode,SetViewportExtEx,SetViewportOrgEx,SetWindowExtEx,
SetWindowOrgEx

SetViewportExtEx	gdi32.dll	95	NT
デバイスコンテキストのビューポートの水平垂直方向範囲設定			

V　C

```
BOOL SetViewportExtEx(
    HDC       hdc,        //デバイスコンテキストのハンドル
    int       nXExtent,   //新しい水平方向のビューポート範囲
    int       nYExtent,   //新しい垂直方向のビューポート範囲
    LPSIZE    lpSize      //オリジナルなビューポート範囲
    );
```

V　B

Declare Function SetViewportExtEx Lib "gdi32" Alias "SetViewportExtEx" (ByVal hdc As Long, ByVal nX As Long, ByVal nY As Long, lpSize As SIZE) As Long

パラメータ

- *hdc*　デバイスコンテキストの識別子。
- *nXExtent*　ビューポートの水平方向範囲をデバイス単位で指定する。
- *nYExtent*　ビューポートの垂直方向範囲をデバイス単位で指定する。
- *lpSize*　SIZE構造体へのポインタ。以前のビューポート範囲(デバイス単位)は、この構造体に格納される。*lpSize*がNULLのときは、何も返されない。

戻り値

正常終了	TRUE
異常終了	FALSE

解　説

SetViewportExtEx関数は、指定された値によりデバイスコンテキストのビューポートの水平方向と垂直方向の範囲を設定する。

次のマッピングモードを設定した場合、SetWindowExtEx関数およびSetViewportExtEx関数への呼び出しが無視される。

MM_HIENGLISH	MM_LOMETRIC
MM_HIMETRIC	MM_TEXT
MM_LOENGLISH	MM_TWIPS

MM_ISOTROPICモードを設定したときには、アプリケーションはSetViewportExtEx関数を呼び出す前にSetWindowExtEx関数を呼び出さなければならない。

参　照

GetViewportExtEx,SetWindowExtEx,SIZE

SetWindowExtEx	gdi32.dll	95	NT
デバイスコンテキストの水平垂直方向の範囲設定			

V C

```
BOOL SetWindowExtEx(
    HDC       hdc,        //デバイスコンテキストのハンドル
    int       nXExtent,   //新しい水平方向のウィンドウ範囲
    int       nYExtent,   //新しい垂直方向のウィンドウ範囲
    LPSIZE    lpSize      //オリジナルなウィンドウ範囲
    );
```

V B

Declare Function SetWindowExtEx Lib "gdi32" Alias "SetWindowExtEx" (ByVal hdc As Long, ByVal nX As Long, ByVal nY As Long, lpSize As SIZE) As Long

パラメータ

- *hdc*　デバイスコンテキストの識別子。
- *nXExtent*　ウィンドウの水平方向範囲を論理単位で指定する。
- *nYExtent*　ウィンドウの垂直方向範囲を論理単位で指定する。
- *lpSize*　SIZE構造体へのポインタ。以前のウィンドウ範囲(論理単位)は、この構造体に格納される。*lpSize*がNULLのときは、何も返されない。

戻り値

正常終了	TRUE
異常終了	FALSE

解説

SetWindowExtEx関数は、指定された値によりデバイスコンテキストのウィンドウの水平方向と垂直方向の範囲を設定する。

次のマッピングモードを設定した場合、SetWindowExtEx関数およびSetViewportExtEx関数への呼び出しが無視される。

MM_HIENGLISH	MM_LOMETRIC
MM_HIMETRIC	MM_TEXT
MM_LOENGLISH	MM_TWIPS

MM_ISOTROPICモードを設定したときには、アプリケーションはSetViewportExtEx関数を呼び出す前にSetWindowExtEx関数を呼び出さなければならない。

参照

GetWindowExtEx,SetViewportExtEx,SIZE

SetWorldTransform	gdi32.dll	95	NT
2次元線分の変換			

V C

```
BOOL SetWorldTransform(
    HDC             hdc,        //デバイスコンテキストのハンドル
    CONST XFORM* lpXform    //変換データのアドレス
    );
```

V B

Declare Function SetWorldTransform Lib "gdi32" Alias "SetWorldTransform" (ByVal hdc As Long, lpXform As xform) As Long

パラメータ

- *hdc*　デバイスコンテキストの識別子。
- *lpXform*　変換データを格納するXFORM構造体へのポインタ。

戻り値

正常終了	TRUE
異常終了	FALSE

解　説

SetWorldTransform関数は、指定されたデバイスコンテキストのワールド空間とページ空間の間に2次元の線の変換を設定する。この変換は、スケール、回転、ひずみを使ったり、グラフィック出力に変えることができる。
ワールド空間の座標(x,y)より、ページ空間の座標の変換(x', y')は以下のアルゴリズムで割り出せる。

x'=x*eM11+y*eM21+eDx,
y'=x*eM12+y*eM22+eDy,

変換マトリックスは以下に示す。

|eM11eM120|
|eM21eM220|
|eDxeDy1|

マッピングモード(カレントウィンドウとビューポイント範囲の原点により定義される)は、単位とスケールを定義するために役立つ。
ワールド変換には、デバイス独自の方法で描くスケールや回転が使用される。デフォルトワールド変換は、0オフセットの内部マトリックスである。
SetWorldTransform関数は、あらかじめSetGraphicsMode関数を呼び出しGM_ADVANCEDが設定されている与えられたデバイスコンテキストのグラフィックモードでなければ失敗する。なおかつ、ワールド変換が最初にSetWorldTransform関数やModifyWorldTransform関数を呼び出し、デフォルトの内部変換に再設定しなければ、デフォルトのGM_COMPATIBLEモードにデバイスコンテキストのグラフィックモードを再設定することはできない。

参　照

GetWorldTransform,ModifyWorldTransform,SetGraphicsMode,SetMapMode,SetViewportExtEx,SetViewportOrgEx,SetWindowExtEx,SetWindowOrgEx,XFORM

Device Context

ChangeDisplaySettings

ChangeDisplaySettings	――――	95	NT

ディスプレイセッティングの変更

V C

```
LONG ChangeDisplaySettings(
    LPDEVMODE  lpDevMode,   //DEVMODE構造体へのポインタ
    DWORD      dwflags      //ディスプレイモード設定フラグ
    );
```

パラメータ

・*lpDevMode*　DEVMODE構造体へのポインタ。dmSizeメンバを利用して、サイズの初期化を行う。指定する値は次の値のいずれかになる。

メンバ	意味
dmBitsPerPel	ビットあたりのピクセル
dmPelsWidth	ピクセルの幅
dmPelsHeight	ピクセルの高さ
dmDisplayFlags	画面モードフラグ
dmDisplayFrequency	リフレッシュレート

上の値をDEVMODEメンバに設定するだけでなく、適切なフラグもdmFieldsメンバにセットしなければならない。フラグは、ディスプレイセッティング変更のために、DEVMODE構造体のどのメンバが使われるかを示す。正しいフラグ設定が行われていない場合には、変更は反映されない。次のフラグのうちの1つ以上を設定する。

フラグ	意味
DM_BITSPERPEL	dmBitsPerPelを利用する
DM_PELSWIDTH	dmPelsWidthを利用する
DM_PELSHEIGHT	dmPelsHeightを利用する
DM_DISPLAYFLAGS	dmDisplayFlagsを利用する
DM_DISPLAYFREQENCY	dmDisplayFrequencyを利用する

*lpDevMode*がNULLのとき、すべての値は現在のレジストリ情報が利用される。動的なモード変化の後にデフォルトのモードへ戻すために最も簡単な方法である。

・*dwflags*　　モードがどのように変更されるかを示す。次の値のいずれかを指定する。

フラグ	意味
0	グラフィックモードは動的に変更される。
CDS_UPDATEREGISTRY	グラフィックモードは動的に変更され、レジストリデータベース、ユーザープロファイルに登録される。
CDS_TEST	CDS_UPDATEREGISTRYが指定されて、グラフィックモードを動的に変更することが可能ならば、情報がレジストリに登録されて、DISP_CHANGE_SUCCESSFULが返される。グラフィックモードを動的に変更することができない場合には、情報がレジストリに登録されて、DISP_CHANGE_RESTARTが返される。

Windows NT：情報がレジストリに登録されなかった場合、グラフィックモードは変更されず、DISP_CHANGE_NOTUPDATEDが返される。

戻り値

次の値のいずれかを返す。

値	意味
DISP_CHANGE_SUCCESSFUL	設定変更が正常に行われた
DISP_CHANGE_RESTART	コンピュータを再起動しなければならない
DISP_CHANGE_BADFLAGS	フラグ設定が正しくない
DISP_CHANGE_FAILED	新しい設定の表示に失敗した
DISP_CHANGE_BADMODE	サポートされない設定を指定した
DISP_CHANGE_NOTUPDATED	WindowsNT：レジストリ情報の更新に失敗した

解　説

EnumDisplaySettings関数から返されたDEVMODE構造体を使うことで、ChangeDisplaySettingsで利用するDEVMODE構造体の設定が有効であることが保証される。表示モードが動的に変更されるときには、WM_DISPLAYCHANGEメッセージは以下のメッセージパラメータによって動作中のすべてのアプリケーションに送られる。

パラメータ	意味
wParam	ピクセルごとの新しいビット数
LOWORD(*lParam*)	新しいピクセル幅
HIWORD(*lParam*)	新しいピクセル高さ

参　照

→CreateDC,→DEVMODE,EnumDisplaySettings,→WM_DISPLAYCHANGE

CreateDC	gdi32.dll	95	NT
デバイスコンテキストの作成			

V C

```
HDC CreateDC(
    LPCTSTR          lpszDriver,   //Windows3.1と同じ関数プロトタイプを使うためだけのもの
    LPCTSTR          lpszDevice,   //サポートしているプリンタの名前を指定
    LPCTSTR          lpszOutput,   //Windows3.1と同じ関数プロトタイプを使うためだけのもの
    CONST DEVMODE    *lpInitData   //初期化情報を含むDEVMODE構造体へのポインタ
    );
```

V B

```
Declare Function CreateDC Lib "gdi32" Alias "CreateDCA" (ByVal lpDriverName As String, ByVal lpDeviceName As String, ByVal lpOutput As String, lpInitData As DEVMODE) As Long
```

パラメータ

- *lpszDriver* NULLで終わる文字列である。通常、ディスプレイドライバには"DISPLAY"、プリンタドライバには"WINSPOOL"を指定する。Win32において、このパラメータは無視される。このパラメータは、Windows 3.1と同じ関数プロトタイプを使うためだけのものである。
- *lpszDevice* NULLで終わる文字列を指すポインタである。サポートしているプリンタの名前を指定する。
- *lpszOutput* NULLで終わる文字列を指すポインタである。Win32において、このパラメータは無視される。このパラメータは、Windows 3.1と同じ関数プロトタイプを使うためだけのものである。
- *lpInitData* デバイスドライバのためのデバイス固有の初期化情報を含むDEVMODE構造体へのポインタである。DocumentProperties関数は、指定されたデバイス用の情報を持つこの構造体を取得する。ユーザーがWindowsのプリントマネージャで指定したデフォルトの初期化情報をデバイスドライバが使用する場合、*lpvInitData*パラメータはNULLでなければならない。

戻り値

正常終了	デバイスコンテキストのハンドル
異常終了	NULL

解　説

指定された名前で示されるデバイスに対するデバイスコンテキストを作成する。

参　照

DocumentProperties,→DEVMODE

DeleteDC	gdi32.dll	95	NT
デバイスコンテキストの削除			

V　C

```
BOOL DeleteDC(
    HDC    hdc    //デバイスコンテキストのハンドル
    );
```

V　B

Declare Function DeleteDC Lib "gdi32" Alias "DeleteDC" (ByVal hdc As Long) As Long

パラメータ

・*hdc*　　削除するデバイスコンテキスト。

戻り値

正常終了	TRUE
異常終了	FALSE

解　説

指定されたデバイスコンテキストを削除する。
アプリケーションは、GetDC関数を呼び出して取得したデバイスコンテキストのハンドルを削除してはならない。この場合、ReleaseDC関数を呼び出してデバイスコンテキストを解放しなければならない。

参　照

→CreateDC,→GetDC,ReleaseDC

GetDC	user32.dll	95	NT
デバイスコンテキストのハンドルを取得			

V C

```
HDC GetDC(
    HWND    hwnd    //デバイスコンテキストを取得するウィンドウ
    );
```

V B

```
Declare Function GetDC Lib "user32" Alias "GetDC" (ByVal hwnd As Long) As Long
```

パラメータ

・*hwnd*　　デバイスコンテキストを取得するウィンドウを識別する。

戻り値

正常終了	デバイスコンテキストのハンドル
異常終了	NULL

解　説

指定されたウィンドウのクライアント領域に対応するデバイスコンテキストのハンドルを取得する。以降、このデバイスコンテキストは、グラフィックデバイスインターフェイス(GDI)関数でクライアント領域内を描画するときに利用可能になる。GetDC関数は、指定されたウィンドウのクラススタイルにより、共通、クラス、またはプライベートのデバイスコンテキストを取得する。共通デバイスコンテキストでは、取得するたびにGetDC関数がデフォルトの属性を割り当てる。クラスコンテキストおよびプライベートコンテキストでは、以前の属性を変更せずにそのまま割り当てる。
共通デバイスコンテキストを使って描画した後でデバイスコンテキストを解放するには、ReleaseDC関数を呼び出さなければならない。クラスデバイスコンテキストやプライベートデバイスコンテキストは解放する必要がない。デバイスコンテキストの数は、利用可能なメモリの量によってのみ制限される。

参　照

ReleaseDC,GetWindowDC

ウィンドウのクラススタイル

共通デバイスコンテキスト
クラス 〃
プライベート 〃

Filled Shape

Ellipse	gdi32.dll	95	NT
楕円の描画			

V　C

```
BOOL Ellipse(
    HDC  hdc          //デバイスコンテキストを識別
    int  nLeftRect    //外接する長方形の左上隅の論理x座標を指定
    int  nTopRect     //外接する長方形の左上隅の論理y座標を指定
    int  nRightRect   //外接する長方形の右下隅の論理x座標を指定
    int  nBottomRect  //外接する長方形の右下隅の論理y座標を指定
    );
```

V　B

```
Declare Function Ellipse Lib "gdi32" Alias "Ellipse" (ByVal hdc As Long, ByVal X1 As Long, ByVal Y1 As Long, ByVal X2 As Long, ByVal Y2 As Long) As Long
```

パラメータ

- *hdc*　デバイスコンテキストのハンドルを指定する。
- *nLeftRect*　外接する長方形の左上隅の論理x座標を指定する。
- *nTopRect*　外接する長方形の左上隅の論理y座標を指定する。
- *nRightRect*　外接する長方形の右下隅の論理x座標を指定する。
- *nBottomRect*　外接する長方形の右下隅の論理y座標を指定する。

戻り値

正常終了	TRUE
異常終了	FALSE

拡張エラー情報を取得するには、GetLastError関数を利用する。

解　説

楕円形を描画する。楕円形の中心は指定された長方形の中心になる。楕円形は現在のペンで描かれ、その内部は現在のブラシで塗りつぶされる。
カレントポジションはこの関数によって使用されず、また変更されることもない。

参　照

Arc,ArcTo

FillRect	user32.dll	95	NT
指定のブラシを使用して、指定された長方形を塗りつぶす			

V　C

```
BOOL FillRect(
    HDC           hDC,     //デバイスコンテキストのハンドル
    CONST RECT    *lprc,   //RECT構造体へのポインタ
    HBRUSH        hbr      //ブラシハンドル
    );
```

V　B

```
Declare Function FillRect Lib "user32" Alias "FillRect" (ByVal hdc As Long, lpRect As RECT, ByVal hBrush As Long) As Long
```

パラメータ

- *hDC*　デバイスコンテキストを識別するハンドルを指定。
- *lprc*　塗りつぶす長方形を含むRECT構造体へのポインタ。
- *hbr*　長方形を塗りつぶすために使用するブラシのハンドルを指定。

戻り値

正常終了	TRUE
異常終了	FALSE

拡張エラー情報はGetLastError関数で取得。

参　照

→CreateHatchBrush,→CreatePatternBrush,→CreateSolidBrush,GetStockObject,
→RECT

Pie	gdi32.dll	95	NT
扇形を描画			

V　C

```
BOOL Pie(
    HDC  hdc,           //デバイスコンテキストを識別するハンドル
    int  nLeftRect,     //外接する四角形左上頂点を示すx座標
    int  nTopRect,      //外接する四角形左上頂点を示すy座標
    int  nRightRect,    //外接する四角形右下頂点を示すx座標
    int  nBottomRect,   //外接する四角形右下頂点を示すy座標
    int  nXRadial1,     //円弧の開始点を示すx座標
    int  nYRadial1,     //円弧の開始点を示すy座標
    int  nXRadial2,     //円弧の終了点を示すx座標
    int  nYRadial2      //円弧の終了点を示すy座標
    );
```

V　B

```
Declare Function Pie Lib "gdi32" Alias "Pie" (ByVal hdc As Long, ByVal X1 As Long, ByVal Y1 As Long, ByVal X2 As Long, ByVal Y2 As Long, ByVal X3 As Long, ByVal Y3 As Long, ByVal X4 As Long, ByVal Y4 As Long) As Long
```

パラメータ

- *hdc*　デバイスコンテキストを識別するハンドル
- *nLeftRect*　外接する四角形左上頂点を示すx座標
- *nTopRect*　外接する四角形左上頂点を示すy座標
- *nRightRect*　外接する四角形右下頂点を示すx座標
- *nBottomRect*　外接する四角形右下頂点を示すx座標
- *nXRadial1*　円弧の開始点を示すx座標
- *nYRadial1*　円弧の開始点を示すy座標
- *nXRadial2*　円弧の終了点を示すx座標
- *nYRadial2*　円弧の終了点を示すy座標

戻り値

正常終了	TRUE
異常終了	FALSE

拡張エラー情報はGetLastError関数で取得。

解　説

Pie関数はパラメータで指定された座標空間に扇形を描画する。座標の指定は外接する長方形を示す2点の座標と円弧の開始と終了の2点により行う。この関数の実行によってカレントポジションが更新されることはない。

参　照

AngleArc,Arc,ArcTo,Chord

Polygon	gdi32.dll	95	NT
多角形を描画			

V C

```
BOOL Polygon(
    HDC           hdc,        //デバイスコンテキストを識別するハンドル
    CONST POINT   *lpPoints,  //多角形の頂点を示すPOINT構造体へのポインタ
    int           nCount      //多角形の頂点の数を示す数値
    );
```

V B

Declare Function Polygon Lib "gdi32" Alias "Polygon" (ByVal hdc As Long, lpPoint As POINTAPI, ByVal nCount As Long) As Long

パラメータ

- *hdc* デバイスコンテキストを識別するハンドル。
- *lpPoints* 多角形の頂点座標を定義したPOINT構造体配列へのポインタ。
- *nCount* POINT構造体で定義された配列に含まれる頂点の数を指定する。ここで指定される値は2以上でなければならない。

戻り値

正常終了	TRUE
異常終了	FALSE

拡張エラー情報はGetLastError関数で取得。

解　説

Polygon関数は多角形を描画する。POINT構造体配列で指定する頂点の始点と終点は、自動的に線分で結ばれ閉じた図形が生成される。この関数によってカレントポジションが更新されることはない。

参　照

GetPolyFillMode,→POINT,Polyline,PolylineTo,PolyPolygon,SetPolyFillMode

Font and Text

DrawText	user32.dll	95	NT
テキストの描画			

V　C

```
int DrawText(
        HDC         hdc         //デバイスコンテキストを識別
        LPCTSTR     lpszStr     //描画する文字列へのポインタ
        int         cchStr      //文字列の文字数を指定
        LPRECT      lprc        //RECT構造体へのポインタ
        UINT        wFormat     //テキストの書式化方法
        );
```

V　B

```
Declare Function DrawText Lib "user32" Alias "DrawTextA" (ByVal hdc As Long, ByVal lpStr As String, ByVal nCount As Long, lpRect As RECT, ByVal wFormat As Long) As Long
```

パラメータ

- *hdc*　デバイスコンテキストを識別する。
- *lpszStr*　描画する文字列を指すポインタである。*cchStr*パラメータが-1のときは、NULLで終わる文字列を指定する。
- *cchStr*　文字列の文字数を指定する。cchStrパラメータが-1のときは、*lpszStr*パラメータがNULLで終わる文字列へのポインタであると解釈され、DrawText関数が自動的に文字数を計算する。
- *lprc*　書式化されたテキストを表示する長方形を含むRECT構造体へのポインタである。
- *wFormat*　テキストの書式を指定する。このパラメータは次の値の組み合わせになる。

値	意味
DT_BOTTOM	テキストを下端揃えにする。この値は、DT_SINGLELINEと組み合わせて指定する。
DT_CALCRECT	長方形の幅と高さを決める。複数行のテキストがあるとき、DrawText関数は*lprc*パラメータが指す長方形の幅を使い、テキストの最下行が収まるように長方形の底辺を広げる。単一行だけのテキストに対しては長方形の右側の辺を調整して、行末の文字が長方形の右側に接するようにする。どちらの場合も、DrawText関数は書式化されたテキストの高さを返すが、テキストの描画は行わない。
DT_CENTER	テキストを水平方向中央に揃える。
DT_EXPANDTABS	タブ文字を展開する。デフォルトのタブ設定は8文字である。
DT_EXTERNALLEADING	行の高さにフォントの外部リーディングを加える。通常、このリーディングはテキスト行の高さに含まれていない。

DT_LEFT	テキストを左揃えにする。
DT_NOCLIP	クリッピングを行わずに描画する。DrawText関数は、DT_NOCLIPを使用したときのほうが多少速く動作する。
DT_NOPREFIX	プリフィックスの処理を行なわない。通常、DrawText関数は、ニーモニックプリフィックス「&」を、その次にくる文字にアンダスコア「_」を付けて表示する疑似命令であると解釈し、「&&」を1つの「&」として表示する。DT_NOPREFIXを指定するとこの処理が行われなくなる。
DT_RIGHT	テキストを右揃えにする。
DT_SINGLELINE	テキストを単一行だけで表示する。キャリッジリターンやラインフィードによってもこの行は分割されない。
DT_TABSTOP	タブストップを設定する。
DT_TOP	テキストを上端揃えにする(単一行のときのみ有効)。
DT_VCENTER	テキストを垂直方向中央揃えにします(単一行のときのみ有効)。
DT_WORDBREAK	ワードブレーク文字を指定する。単語が*lprc*パラメータにより指定された長方形の端を越えてしまうときには、自動的に単語間で行が分割される。キャリッジリターンとラインフィードの組み合わせによっても行が分割される。

戻り値

正常終了	テキストの高さ

解　説

指定された長方形の領域に整形したテキストを描画する。この関数は、指定された方法(タブの展開、文字の位置合わせ、行の折り返しなど)にしたがってテキストを整形する機能がある。
DrawText関数は、デバイスコンテキストで選択されたフォント、テキストの色、および背景色を使ってテキストを描画する。

参　照

GrayString,TabbedTextOut,→TextOut,→RECT

GetAspectRatioFilterEx	gdi32.dll	95	NT
現在のアスペクト比フィルタを取得			

V C

```
BOOL GetAspectRatioFilterEx(
        HDC       hdc,            //デバイスコンテキストを識別するハンドル
        LPSIZE    lpAspectRatio   //アスペクト比フィルタを識別するSIZE構造体へ
                                    のポインタ
        );
```

V B

```
Declare Function GetAspectRatioFilterEx Lib "gdi32" Alias "GetAspectRatioFilterEx"
(ByVal hdc As Long, lpAspectRatio As SIZE) As Long
```

パラメータ

・*hdc*　デバイスコンテキストを識別するハンドル。
・*lpAspectRatio*　アスペクト比フィルタを識別するSIZE構造体へのポインタ。

戻り値

正常終了	TRUE
異常終了	FALSE

拡張エラー情報はGetLastError関数で取得。

解　説

アスペクト比とは指定されたデバイスの縦横比を示すものである。

参　照

SetMapperFlags,→SIZE

GetFontData	gdi32.dll	95	NT
TrueTypeフォントファイルから、フォントメトリックデータを取得			

V C

```
int GetFontData(
        HDC        hdc,         //デバイスコンテキストを識別
        DWORD      dwTable,     //メトリックテーブルの名称を指定
        DWORD      dwOffset,    //テーブルの先頭からのオフセット位置
        LPVOID     lpvBuffer,   //フォントを受け取るバッファへのポインタ
        DWORD      cbData       //取得する情報の長さをバイトで指定
        );
```

V B

```
Declare Function GetFontData Lib "gdi32" Alias "GetFontDataA" (ByVal hdc As Long, ByVal dwTable As Long, ByVal dwOffset As Long, lpvBuffer As Any, ByVal cbData As Long) As Long
```

パラメータ

- *hdc* デバイスコンテキストを識別するハンドル。
- *dwTable* メトリックテーブルの名前を指定。このパラメータには、TrueTypeフォントファイル仕様書の中で解説されている、メトリックテーブルの1つを指定できる。パラメータが0のときは、フォントファイルの先頭から情報が取得される。
- *dwOffset* 情報の取得を開始する、テーブルの先頭からのオフセット位置を指定。このパラメータが0のときは、*dwTable*パラメータで指定されたテーブルの先頭から情報が取得される。パラメータの値がテーブルのサイズを超えた場合には、GetFontDataは0を返す。
- *lpBuffer* フォント情報を受け取るバッファへのポインタ。この値がNULLのときは、*dwTable*パラメータで指定されたフォントデータを格納するために必要なバッファのサイズを返す。
- *cbData* 取得する情報の長さをバイト単位で指定。このパラメータが0のとき、GetFontData関数は*dwTable*パラメータで指定されたデータのサイズを返す。

戻り値

正常終了	*lpBuffer*パラメータに返されるバイト数
異常終了	-1

解　説

アプリケーションは、GetFontData関数を使用してTrueTypeフォントを文書とともに保存できる。これを行うためにフォントが埋め込み可能かどうかを判断してから*dwTable*、*dwOffset*、*cbData*の各パラメータに0を指定して全フォントファイルを取得する。

TOUTLINETEXTMETRIC構造体のotmfsTypeメンバをチェックすることで、フォントが埋め込み可能かどうかを判断できる。otmfsTypeのビット1がセットされているときには、フォントの埋め込みができない。ビット1をクリアすることでフォントが埋め込み可能になる。ビット2をセットすると、フォントの埋め込みが読み取り専用属性になる。アプリケーションがこの関数を使用してTrueType以外のフォントの情報を取得しようとした場合、GetFontData関数は-1を返す。

参照 GetOutlineTextMetrics

PolyTextOut | gdi32.dll | | NT

複数の文字列を表示

V C

```
BOOL PolyTextOut(
    HDC                 hdc,       //デバイスコンテキストを識別するハンドル
    CONST POLYTEXT      *pptxt,    //文字列配列を識別するPOLYTEXT構造体
                                     へのポインタ
    int                 cStrings   //pptxtパラメータに指定されたPOLYTEXT
                                     構造体の数
    );
```

V B

```
Declare Function PolyTextOut Lib "gdi32" Alias "PolyTextOutA" (ByVal hdc As Long,
pptxt As POLYTEXT, cStrings As Long) As Long
```

パラメータ

- *hdc* デバイスコンテキストを識別するハンドルを指定する。
- *pptxt* 文字列を記述しているPOLYTEXT構造体の配列へのポインタ。配列は、1つの文字列ごとに1つの構造体を含んでいる。
- *cStrings* *pptxt*パラメータに指定されたPOLYTEXT構造体の数を指定する。

戻り値

正常終了	TRUE
異常終了	FALSE

拡張エラー情報はGetLastError関数で取得する。

解説

個々のPOLYTEXT構造体は、テキストを位置合わせするためにWindowsが用いる参照ポイントの座標を含んでいる。アプリケーションは、参照ポイントがどのように使われて、SetTextAlign関数を呼ぶかを指定できる。アプリケーションは、指定されたデバイスコンテキストのために、GetTextAlign機能を呼んで設定する現在のテキストを決定できる。単一テキストを表示するためには、ExtTextOut関数を呼ぶべきである。

参照 ExtTextOut,→GetTextAlign,→POLYTEXT,→SetTextAlign

SetTextAlign	gdi32.dll	95	NT
文字列をデバイスに配置する			

V C

```
UINT SetTextAlign(
    HDC  hdc,    //デバイスコンテキストを識別するハンドル
    UINT fMode   //配置フラグ
    );
```

V B

```
Declare Function SetTextAlign Lib "gdi32" Alias "SetTextAlign" (ByVal hdc As Long, ByVal wFlags As Long) As Long
```

パラメータ

- *hdc* デバイスコンテキストを識別するハンドル。
- *fMode* 次の値のいずれかを指定してテキストの配置を指定する。水平、垂直方向に関する指定を1つ、さらにカレントポジションを変更する2つのフラグから1つを選択する。

値	意味
TA_BASELINE	参照ポイントはテキストのベースラインにある。
TA_BOTTOM	参照ポイントは隣接長方形の底辺にある。
TA_TOP	参照ポイントは隣接長方形の一番上の辺にある。
TA_CENTER	参照ポイントは隣接長方形の水平中央に位置合わせされる。
TA_LEFT	参照ポイントは隣接長方形の左辺にある。
TA_RIGHT	参照ポイントは隣接長方形の右辺にある。
TA_NOUPDATECP	カレントポジションは個々のテキスト出力関数の呼び出しの後に更新されない。参照ポイントはテキスト出力関数に渡される。
TA_RTLREADING	Windows 95：テキストは右から左に流される。これは、選ばれたフォントがヘブライもしくはアラビア語のときにだけ適用される。
TA_UPDATECP	カレントポジションは個々のテキスト出力関数の呼び出しの後に更新される。カレントポジションは参照ポイントとして使われる。

現在のフォントが漢字の場合には、TA_BASELINEとTA_CENTERの代わりに次の値を指定する。

値	意味
VTA_BASELINE	参照ポイントはテキストのベースラインにある。
VTA_CENTER	参照ポイントは隣接長方形の垂直中央に位置合わせされる。

デフォルトの値はTA_LEFT,TA_TOP,TA_NOUPDATECPである。

戻り値

正常終了	以前のテキスト配置設定
異常終了	GDI_ERROR

拡張エラー情報はGetLastError関数で取得する。

解説

TextOutとExtTextOut関数は、表示または他の機器と同時にテキストの文字列を配置するために、テキスト整列フラグを使う。フラグは、参照ポイントとテキストと接している長方形の間の関係を指定する。参照ポイントは、テキスト出力関数に渡されたカレントポジションまたはポイントである。

参照

ExtTextOut,→TextOut

SetTextColor	gdi32.dll	95	NT
テキスト色を設定する			

VC

```
COLORREF SetTextColor(
    HDC         hdc,      //デバイスコンテキストを識別するハンドル
    COLORREF    crColor   //テキストの色
    );
```

VB

```
Declare Function SetTextColor Lib "gdi32" Alias "SetTextColor" (ByVal hdc As Long,
ByVal crColor As Long) As Long
```

パラメータ

- *hdc*　デバイスコンテキストを識別するハンドル。
- *crColor*　テキストの色を指定する。

戻り値

正常終了	以前のテキストの色
異常終了	CLR_INVALID

拡張エラー情報はGetLastError関数で取得。

解説

デバイスが指定された色を表示できない場合には、その色に最も近い色がテキストの色として設定される。文字の背景色はSetBkColor関数を使用して指定する。

参照

→BitBlt,ExtTextOut,→RGB,SetBkColor,→StretchBlt,→TextOut

TextOut	gdi32.dll	95	NT
現在のフォントを使用して、文字列を書き込む			

V C

```
BOOL TextOut(
    HDC      hdc,       // デバイスコンテキストを識別するハンドル
    int      nXStart,   // 文字列を開始する論理x座標
    int      nYStart,   // 文字列を開始する論理y座標
    LPCTSTR  lpString,  // 描画するテキストを格納した文字列へのポインタ
    int      cbString   // 描画する文字列のバイト数
    );
```

V B

```
Declare Function TextOut Lib "gdi32" Alias "TextOutA" (ByVal hdc As Long, ByVal x As Long, ByVal y As Long, ByVal lpString As String, ByVal nCount As Long) As Long
```

パラメータ

- *hdc*　デバイスコンテキストを識別するハンドル。
- *nXStart*　文字列を開始する論理x座標を指定する。
- *nYStart*　文字列を開始する論理y座標を指定する。
- *lpString*　描画するテキストを格納した文字列へのポインタを指定する。このパラメータで指定される文字列には全角文字を含めることが可能。
- *cbString*　描画する文字列のバイト数を指定する。

戻り値

正常終了	TRUE
異常終了	FALSE

拡張エラー情報はGetLastError関数で取得。

解　説

描画される文字の原点は、文字セルの左上隅になる。デフォルトではTextOut関数により現在の位置が使われたり更新されたりすることはない。アプリケーションでTextOut関数を呼び出したときにカレントポジションを更新しなければならない場合には、*Flags*パラメータをTA_UPDATECPにセットしてSetTextAlign関数を呼び出す。Windowsはこのフラグがセットされると以後のTextOut関数呼び出しで、*nXStart*、*nYStart*パラメータを無視して、カレントポジションを参照する。

参　照

GetTextAlign,SetTextAlign,TabbedTextOut

Metafile

CreateEnhMetaFile	gdi32.dll	95	NT
拡張メタファイルを作成			

V　C

```
HDC CreateEnhMetaFile(
    HDC            hdcRef,        //参照するデバイスコンテキストを識別するハンドル
    LPCTSTR        lpFilename,    //ファイル名を格納した文字列へのポインタ
    CONST RECT *lpRect,           //隣接長方形を識別するRECT構造体へのポインタ
    LPCTSTR        lpDescription  //任意の記述文字列へのポインタ
);
```

V　B

```
Declare Function CreateEnhMetaFile Lib "gdi32" Alias "CreateEnhMetaFileA" (ByVal hdcRef As Long, ByVal lpFileName As String, lpRect As RECT, ByVal lpDescription As String) As Long
```

パラメータ

- *hdcRef*　拡張メタファイルの参照デバイスコンテキストを識別するハンドル。
- *lpFilename*　拡張メタファイルが作成するファイル名へのポインタ。このパラメータがNULLのときは、拡張メタファイルはメモリ上に作成され、DeleteEnhMetaFile関数を使って削除されるときは、その内容が失われる。
- *lpRect*　拡張メタファイルに保存されるグラフィックの寸法（0.01ミリメートル単位）を指定するRECT構造体へのポインタ。
- *lpDescription*　作成した拡張メタファイルのタイトル名を指定する文字列へのポインタ。

戻り値

正常終了	拡張メタファイルのためのデバイスコンテキストを識別するハンドル
異常終了	NULL

解　説

CreateEnhMetaFile関数は拡張メタファイルフォーマットのデバイスコンテキストを生成する。作成されたデバイスコンテキストはデバイスに依存しない形式のグラフィックとして利用可能である。

参　照

CloseEnhMetaFile,DeleteEnhMetaFile,GetEnhMetaFileDescription,
GetEnhMetaFileHeader,GetWinMetaFileBits,PlayEnhMetaFile,→RECT

CreateMetaFile	gdi32.dll	95	NT
メタファイルのデバイスコンテキストを作成			

V C

```
HDC CreateMetaFile(
    LPCSTR    lpszFile    //作成するメタファイルのMS-DOSファイル名
    );
```

V B

Declare Function CreateMetaFile Lib "gdi32" Alias "CreateMetaFileA" (ByVal lpString As String) As Long

パラメータ

・*lpszFile*　作成するメタファイルに付けるMS-DOSのファイル名を指定するNULLで終わる文字列を指すポインタである。このパラメータがNULLのときには、メモリメタファイル用のデバイスコンテキストを返す。この文字列には、全角文字を含めることが可能である。

戻り値

正常終了	デバイスコンテキストのハンドル
異常終了	NULL

解　説

Windows形式メタファイルのデバイスコンテキストを作成する。
Win32アプリケーションについては、CreateEnhMetaFile関数を使用するのが望ましい。

参　照

CloseMetaFile,DeleteMetaFile,GetTextColor,PolyBezier,→SelectClipRgn

GetEnhMetaFile	user32.dll	95	NT
拡張メタファイルハンドルの取得			

V　C

```
HENHMETAFILE GetEnhMetaFile(
    LPCTSTR  lpszMetaFile  //メタファイルを格納したファイル名を示す文字列へのポインタ
);
```

V　B

Declare Function GetEnhMetaFile Lib "gdi32" Alias "GetEnhMetaFileA" (ByVal lpszMetaFile As String) As Long

パラメータ

・*lpszMetaFile*　　拡張メタファイルを格納したファイルの名称を格納したNULLで終わる文字列へのポインタ。

戻り値

正常終了	拡張メタファイルのハンドル
異常終了	NULL

解　説

拡張メタファイルハンドルが不要になった場合には、DeleteEnhMetaFile関数でハンドルを削除しなければならない。GetEnhMetaFile関数で処理するためには、ウィンドウズ型式メタファイルは、拡張メタファイル形式に変換されなければならない。ファイルを変換するためには、SetWinMetaFileBits関数を使う。
Windows 95：拡張メタファイルのタイトル記述文字列の最大長は、16,384バイトである。

参　照

DeleteEnhMetaFile,→GetEnhMetaFile,SetWinMetaFileBits

GetMetaFile	gdi32.dll	95	NT
Windowsメタファイルを作成			

V C

```
HMETAFILE GetMetaFile(
        LPCTSTR        lpszFile    //メタファイルのファイル名称
        );
```

V B

```
Declare Function GetMetaFile Lib "gdi32" Alias "GetMetaFileA" (ByVal lpFileName As String) As Long
```

パラメータ

・*lpszFile* メタファイルを識別するNULLで終わるファイル名へのポインタ。

戻り値

正常終了	メタファイルへのハンドル
異常終了	NULL

参　照

DeleteMetaFile,→GetEnhMetaFile,SetWinMetaFileBits

Path

BeginPath	gdi32.dll	95	NT
パスブラケットの開始			

V　C

```
BOOL BeginPath(
    HDC  hdc    //デバイスコンテキストを識別するハンドル
    );
```

V　B

```
Declare Function BeginPath Lib "gdi32" Alias "BeginPath" (ByVal hdc As Long) As Long
```

パラメータ

・*hdc*　　デバイスコンテキストを識別するハンドルを指定する。

戻り値

正常終了	TRUE
異常終了	FALSE

拡張エラー情報はGetLastError関数で取得する。

解　説

アプリケーションは、パスブラケットが開いた後でパスにあるポイントを定義するため、GDI関数を利用可能になる。アプリケーションは、EndPath関数で開いているパスブラケットを閉じることができる。
デバイスコンテキストのためにアプリケーションがBeginPath関数を呼び出すときには、それ以前のパスはすべて破棄される。

参　照

→EndPath,FillPath,→PathToRegion,SelectClipPath,StrokeAndFillPath,StrokePath,WidenPath

CloseFigure	gdi32.dll	95	NT
パス中の図形を閉じる			

V C

```
BOOL CloseFigure(
    HDC    hdc    //デバイスコンテキストを識別するハンドル
    );
```

V B

Declare Function CloseFigure Lib "gdi32" Alias "CloseFigure" (ByVal hdc As Long) As Long

パラメータ

・*hdc*　　閉じられるデバイスコンテキストを識別するハンドルを指定する。

戻り値

正常終了	TRUE
異常終了	FALSE

拡張エラー情報はGetLastError関数で取得する。

解　説

CloseFigure関数は、図の最初のポイントと現在のポジションを結んで図を閉じる。このとき、ラインジョインスタイルを使ってラインを接続する。CloseFigure関数は、指定されたデバイスコンテキストに、開いているパスブラケットがあるときには何もしない。この機能を使って明示的に閉じられない限り、パスは開いている（たとえ現在のポイントと図の起点が同じでも、パスは開いているかもしれない）。

参　照

→BeginPath,→EndPath,ExtCreatePen,→LineTo,→MoveTo

EndPath	gdi32.dll	95	NT
パスブラケットの終了			

V C

```
BOOL EndPath(
    HDC    hdc    //デバイスコンテキストを識別するハンドル
    );
```

V B

Declare Function EndPath Lib "gdi32" Alias "EndPath" (ByVal hdc As Long) As Long

パラメータ

・*hdc*　　デバイスコンテキストを識別するためのハンドルを指定する。

戻り値

正常終了	TRUE
異常終了	FALSE

拡張エラー情報はGetLastError関数で取得する。GetLastError関数は次のいずれかのコードを返す。

ERROR_CAN_NOT_COMPLETE,ERROR_INVALID_PARAMETER

参　照

→BeginPath

GetPath	gdi32.dll	95	NT
パス内の直線と曲線を取得			

V　C

```
int GetPath(
    HDC        hdc,       //デバイスコンテキストを識別するハンドル
    LPPOINT    lpPoints,  //POINT構造体へのポインタ
    LPBYTE     lpTypes,   //頂点のタイプを指定する配列へのポインタ
    int        nSize      //パス中の頂点の数
    );
```

V　B

```
Declare Function GetPath Lib "gdi32" Alias "GetPath" (ByVal hdc As Long, lpPoint As POINTAPI, lpTypes As Byte, ByVal nSize As Long) As Long
```

パラメータ

・*hdc*　デバイスコンテキストを識別するためのハンドルを指定する。

・*lpPoints*　ライン末端と曲線コントロールポイントを含むPOINT構造体配列へのポインタ。

・*lpTypes*　頂点タイプが置かれるバイトの配列へのポインタ。次のいずれかの値を指定する。

タイプ	意味
PT_MOVETO	*lpPoints*パラメータが開いた図形を指定する
PT_LINETO	*lpPoints*パラメータで指定されるライン終端が次のラインの始点である
PT_BEZIERTO	*lpPoints*パラメータで指定される曲線コントロールがベジェ曲線を表す

PT_LINETO、PT_BEZIERTOの値は、図が閉じられるべきであることを示すために、次の値と結合できる。

フラグ	意味
PT_CLOSEFIGURE	ラインまたは曲線が引かれた後に、図が自動的に閉じられることを指定する。図は、最後のPT_MOVETOと一致しているポイントにラインをラインまたは曲線末端から引くことによって閉じられる。

・*nSize*　*lpPoints*によりさし示されたPOINT構造体配列の数を指定する。

戻り値

nSiZE=0以外	列挙された頂点の数
nSiZE=0	パス中の全頂点の数
異常終了	-1

拡張エラー情報はGetLastError関数で取得する。GetLastError関数は次のいずれかのコードを返す。

ERROR_CAN_NOT_COMPLETE,ERROR_INVALID_PARAMETER,
ERROR_BUFFER_OVERFLOW

解説

*hdc*パラメータにより識別されたデバイスコンテキストは、閉じられたパスを含む。パスの頂点は論理的な座標で戻される。頂点はデバイスコンテキストのパスに保存される。GetPathは現在の変化の逆の手順を使ってデバイスコンテキストから頂点を論理的な座標に交換する。FlattenPath関数は、パスのすべての曲線をラインセグメントに変換するためにGetPathの前に呼ばれる。

参照

→FlattenPath,→POINT,→PolyDraw,WidenPath

PathToRegion | gdi32.dll | 95 | NT

パスからリージョンを作成

VC

```
HRGN PathToRegion(
    HDC      hdc      //デバイスコンテキストを識別するハンドル
    );
```

VB

Declare Function PathToRegion Lib "gdi32" Alias "PathToRegion" (ByVal hdc As Long) As Long

パラメータ

・*hdc*　閉じたパスを含むデバイスコンテキストを識別するハンドルを指定する。

戻り値

正常終了	リージョンを識別するハンドル
異常終了	0

拡張エラー情報はGetLastError関数で取得する。GetLastError関数は次のいずれかのコードを返す。

ERROR_CAN_NOT_COMPLETE,ERROR_INVALID_PARAMETER,
ERROR_NOT_ENOUGH_MEMORY

解説

*hdc*パラメータにより識別されたデバイスコンテキストは、閉じられたパスを含む。PathToRegion関数がパスからリージョンを作成した後に、Windowsは指定されたデバイスコンテキストからパスを破棄する。

参照

→BeginPath,→EndPath,SetPolyFillMode

Region

FillRgn	gdi32.dll	95	NT
指定されたリージョンをブラシパターンで塗りつぶす			

V C

```
BOOL FillRgn(
    HDC      hdc,    //デバイスコンテキストを識別
    HRGN     hrgn,   //デバイス単位で指定した塗りつぶすリージョンを識別
    HBRUSH   hbr     //塗りつぶしに使用するブラシを識別
    );
```

V B

```
Declare Function FillRgn Lib "gdi32" Alias "FillRgn" (ByVal hdc As Long, ByVal hRgn As Long, ByVal hBrush As Long) As Long
```

パラメータ

- *hdc*　デバイスコンテキストを識別する
- *hrgn*　関数によって塗りつぶされるリージョンを指定する。リージョンの座標はデバイス座標を用いて指定する。
- *hbr*　リージョンの塗りつぶしに利用されるブラシを識別する。

戻り値

正常終了	TRUE
異常終了	FALSE

参照

→CreateBrushIndirect,→CreateDIBPatternBrush,→CreateHatchBrush,
→CreatePatternBrush,→CreateSolidBrush,→PaintRgn

GetRgnBox	gdi32.dll	95	NT
リージョンの境界長方形の座標を取得			

V C

```
int GetRgnBox(
        HRGN     hrgn,  //リージョンを識別するハンドル
        LPRECT   lprc   //境界長方形の座標を受け取るRECT構造体へのポインタ
```

V B

Declare Function GetRgnBox Lib "gdi32" Alias "GetRgnBox" (ByVal hRgn As Long, lpRect As RECT) As Long

パラメータ

・*hrgn*　リージョンを識別するためのハンドル。
・*lprc*　境界長方形の座標を受け取るためのRECT構造体へのポインタを指定する。

戻り値

関数が正常に終了した場合、リージョンの状態を表す以下の値が返される。

NULLREGION	リージョンが空
SIMPLEREGION	リージョンの境界が重なっていない
COMPLEXREGION	リージョンの境界が重なっている

パラメータで指定されたリージョンが存在しない場合には0が返される。

参照

→RECT

PaintRgn	gdi32.dll	95	NT
選択されているブラシパターンでリージョンを塗りつぶす			

V C

```
BOOL PaintRgn(
        HDC    hdc,     //デバイスコンテキストを識別するハンドル
        HRGN   hrgn     //塗りつぶされるリージョンを識別するハンドル
        );
```

V B

Declare Function PaintRgn Lib "gdi32" Alias "PaintRgn" (ByVal hdc As Long, ByVal hRgn As Long) As Long

パラメータ

・*hdc*　デバイスコンテキストを識別するハンドルを指定する。
・*hrgn*　関数によって塗りつぶされるリージョンを識別するハンドルを指定する。リージョンの座標はデバイス座標によって指定される。

戻り値

正常終了	TRUE
異常終了	FALSE

参照

→FillRgn

2. 6 System Service
システムサービス

メモリ管理をはじめ構造化例外処理などのエラーハンドリングと、スレッドやパイプの処理を担うシステムサービス関数群である。レジストリ操作とスレッド、デスクトップに関する操作は32ビットアプリケーション開発の中でも重要な位置をしめる部分である。

Accessibitity

GetSystemMetrics	user32.dll	95	NT
システムメトリックに関する情報を取得			

V　C

```
int GetSystemMetrics(
    int          nIndex     //取得したいシステムメトリックを指定
    );
```

V　B

```
Declare Function GetSystemMetrics Lib "user32" Alias "GetSystemMetrics" (ByVal nIndex As Long) As Long
```

パラメータ

・*nIndex*　取得したいシステムメトリックを以下の値で指定する。

値	意味
SM_CMOUSEBUTTONS	マウスのボタン数
SM_CXBORDER	サイズ変更できないウィンドウ枠の幅
SM_SM_CYBORDER	サイズ変更できないウィンドウ枠の高さ
SM_CXCURSOR	カーソルの幅
SM_CYCURSOR	カーソルの高さ
SM_CXDLGFRAME	ダイアログウィンドウスタイルのウィンドウ枠の幅
SM_CYDLGFRAME	ダイアログウィンドウスタイルのウィンドウ枠の高さ
SM_CXDOUBLECLK	ダブルクリックで最初のクリックが実行されたときの周辺の長方形の幅
SM_CYDOUBLECLK	ダブルクリックで最初のクリックが実行されたときの周辺の長方形の高さ
SM_CXFRAME	サイズ変更枠を持つウィンドウ枠の幅

SM_CYFRAME	サイズ変更枠を持つウィンドウ枠の高さ
SM_CXFULLSCREEN	フルスクリーンウィンドウのクライアント領域の幅
SM_CYFULLSCREEN	フルスクリーンウィンドウのクライアント領域の高さ
SM_CXHSCROLL	水平スクロールバーの矢印ビットマップの幅
SM_CYHSCROLL	水平スクロールバーの矢印ビットマップの高さ
SM_CXHTHUMB	水平スクロールバーのスクロールボックス（つまみ）の幅
SM_CXICON	アイコンの幅
SM_CYICON	アイコンの高さ
SM_CXICONSPACING	アイコンを整列させた際に利用されるアイコン周辺の長方形の幅
SM_CYICONSPACING	アイコンを整列させた際に利用されるアイコン周辺の長方形の高さ
SM_CXMIN	ウィンドウの幅の最小値
SM_CYMIN	ウィンドウの高さの最小値
SM_CXMINTRACK	ウィンドウのトラッキング幅の最小値
SM_CYMINTRACK	ウィンドウのトラッキング高さの最小値
SM_CXSCREEN	スクリーン領域の幅
SM_CYSCREEN	スクリーン領域の高さ
SM_CXSIZE	タイトルバーに含まれるビットマップの幅
SM_CYSIZE	タイトルバーに含まれるビットマップの高さ
SM_CXVSCROLL	垂直スクロールバーの矢印ビットマップの幅
SM_CYVSCROLL	垂直スクロールバーの矢印ビットマップの高さ
SM_CYCAPTION	ウィンドウタイトルの高さ
SM_CYKANJIWINDOW	漢字ウィンドウの高さ
SM_CYMENU	単一行で構成されるメニューバーの高さ
SM_CYVTHUMB	水平スクロールバーのスクロールボックス（つまみ）の高さ
SM_DBCSENABLED	USER.EXEの現在のバージョンが2バイト文字を使用している場合にはTRUEもしくは0以外の値になる。それ以外の場合にはFALSEもしくは0になる
SM_DEBUG	USER.EXEの現在のバージョンがデバッグバージョンの場合にはTRUEもしくは0以外になる。それ以外の場合はFALSEもしくは0になる
SM_MENUDROPALIGNMENT	ポップアップメニューの位置。TRUEもしくは0以外の場合にはポップアップメニューはメニューバー項目の左側に揃えられる。FALSEもしくは0の場合には右に揃えられる
SM_MOUSEPRESENT	マウスがインストールされている場合はTRUEもしくは0以外の値が返される。インストールされていない場合にはFALSEもしくは0である

SM_PENWINDOWS	Microsoft Windows for Penがインストールされている場合には、TRUEもしくは0以外の値になる。それ以外の場合にはFALSEもしくは0である
SM_SWAPBUTTON	マウスの左右ボタンの機能が交換されている場合にはTRUEもしくは0以外。これ以外の場合にはFALSEもしくは0となる
SM_CLEANBOOT	Windows 95。正常起動した場合には0。SafeBootモードの場合には1。SafeBoot（ネットワーク）の場合には2が設定される
SM_CMETRICS	Windows 95 のシステムメトリックフラグの数
SM_CXDRAG	Windows 95。ドラッグ操作可能な長方形領域の幅
SM_CYDRAG	Windows 95。ドラッグ操作可能な長方形領域の高さ
SM_CXEDGE	Windows 95。3Dフレームの幅
SM_CYEDGE	Windows 95。3Dフレームの高さ
SM_CXFIXEDFRAME	Windows 95。サイズ固定ウィンドウ（キャプション有り）の幅
SM_CYFIXEDFRAME	Windows 95。サイズ固定ウィンドウ（キャプション有り）の高さ
SM_CXMAXIMIZED	Windows 95。最大化ウィンドウの幅
SM_CYMAXIMIZED	Windows 95。最大化ウィンドウの高さ
SM_CXMAXTRACK	Windows 95。ウィンドウトラッキング幅の最大値
SM_CYMAXTRACK	Windows 95。ウィンドウトラッキング高さの最大値
SM_CXMENUCHECK	Windows 95。標準のメニューチェック状態ビットマップの幅
SM_CYMENUCHECK	Windows 95。標準のメニューチェック状態ビットマップの高さ
SM_CXMENUSIZE	Windows 95。メニューバーのボタンの幅
SM_CYMENUSIZE	Windows 95。メニューバーのボタンの高さ
SM_CXMINIMIZED	Windows 95。ウィンドウの幅の最小値
SM_CYMINIMIZED	Windows 95。ウィンドウの高さの最小値
SM_CXMINSPACING	Windows 95。アイコンを整列させた際に利用されるアイコン周辺の長方形の幅
SM_CYMINSPACING	Windows 95。アイコンを整列させた際に利用されるアイコン周辺の長方形の高さ
SM_CXSIZEFRAME	Windows 95。サイズ変更枠を持つウィンドウ枠の幅
SM_CYSIZEFRAME	Windows 95。サイズ変更枠を持つウィンドウ枠の高さ
SM_CXSMICON	Windows 95。ウィンドウタイトルバーに表示される小さなアイコンの幅
SM_CYSMICON	Windows 95。ウィンドウタイトルバーに表示される小さなアイコンの高さ
SM_CXSMSIZE	Windows 95。ウィンドウタイトルのボタンの幅
SM_CYSMSIZE	Windows 95。ウィンドウタイトルのボタンの高さ
SM_CYSMCAPTION	Windows 95。ウィンドウタイトルの高さ
SM_MIDEASTENABLED	Windows 95。ヘブライ語およびアラビア語文字セットが利用可能な場合にはTRUE

SM_NETWORK	Windows 95。ネットワークがインストールされている場合には先頭ビットがオンになる。他のビットは将来の拡張のために予約されている
SM_SECURE	Windows 95。セキュリティがかけられている場合にはTRUE、それ以外の場合にはFALSEとなる
SM_SLOWMACHINE	Windows 95。使用中のコンピュータシステムのプロセッサーがローエンド（遅い）の場合にはTRUE、それ以外の場合にはFALSEとなる

戻り値

パラメータで要求されたシステムメトリックの値を返す。

解　説

GetSystemMetrics関数の戻り値は利用しているスクリーンタイプにより変化する。SM_ARRANGEパラメータを指定してWindowsが最小化時のアイコンをどのように整列するかを取得するには次のスタート位置から1つを選ぶ。

値	意味
ARW_BOTTOMLEFT	左下を整列開始位置にする（初期値）
ARW_BOTTOMRIGHT	右下を整列開始位置にする
ARW_HIDE	最小化時に表示しない
ARW_TOPLEFT	左上を整列開始位置にする
ARW_TOPRIGHT	右上を整列開始位置にする

整列の方向については次から1つを選ぶ

値	意味
ARW_DOWN	上から下に整列させる
ARW_LEFT	左から右に整列させる
ARW_RIGHT	右から左に整列させる
ARW_UP	下から上に整列させる

参　照

GetSysColor,SystemParametersInfo

Data Decompression Library

LZClose	lz32.dll	95	NT
ファイルをクローズ			

V　C

```
void LZClose(
    INT        hFile      //クローズしたいファイルのファイルハンドルを指定
    );
```

V　B

Declare Sub LZClose Lib "lz32.dll" Alias "LZClose" (ByVal hfFile As Long)

パラメータ

・*hFile*　　クローズしたい任意のファイルを指定するファイルハンドル。

解　説

LZClose関数はファイルのクローズ処理を実行する。クローズするためにパラメータで渡されるファイルハンドルはLZOpenFile関数によってオープンされたファイルでなければならない。CreateFileやOpenFile関数によってオープンされたファイルハンドルをLZClose関数でクローズしようとした場合はエラーが返される。COMPRESS.EXE（Microsoftファイル圧縮ユーティリティ）で圧縮されたファイルをLZOpenFile関数を利用してオープンしていた場合には、LZClose関数を利用することで、それまでのグローバルヒープが解放される。

参　照

CreateFile,→LZOpenFile,OpenFile

LZCopy	lz32.dll	95	NT
ファイルをコピーして圧縮されていればそれを展開			

V　C

```
LONG LZCopy(
    int        hfSource,  //コピー元ファイルのハンドル
    int        hfDest     //コピー先ファイルのハンドル
    );
```

V　B

Declare Function LZCopy Lib "lz32.dll" Alias "LZCopy" (ByVal hfSource As Long, ByVal hfDest As Long) As Long

パラメータ

・*hfSource*　　コピー元のファイルを識別するハンドル。

・*hfSource*　　コピー先のファイルを識別するハンドル。変換されるために使われるクライアント領域を持つウィンドウの識別子。

戻り値

正常終了時にはファイルサイズをバイト単位で返す。
異常終了時には次のコードのいずれかを返す。

LZERROR_BADINHANDLE	コピー元ファイルのハンドルが無効
LZERROR_BADOUTHANDLE	コピー先ファイルのハンドルが無効
LZERROR_GLOBALLOC	上限数を上回るファイルを既にオープンしている。もしくはローカルメモリの取得に失敗した
LZERROR_GLOBLOCK	圧縮ファイルがロックされている
LZERROR_READ	コピー元のファイル形式が無効

解　説

LZCopy関数はパラメータで指定されたファイルを任意のファイルとしてコピーする。コピー元ファイルがMicrosoft圧縮ユーティリティによって圧縮されている場合は、コピー操作にさきがけて解凍作業も実行する。LZCopyのパラメータで指定するファイルハンドルはLZInit、LZOpenFile関数で取得したファイルハンドルでなければならない。

参　照

CopyLZFile,LZInit,→LZOpenFile

LZOpenFile	lz32.dll	95	NT
圧縮および非圧縮のファイルをオープン			

V　C

```
int LZOpenFile(
    LPTSTR       lpFileName,    //ファイル名を格納する文字列へのポインタ
    LPOFSTRUCT   lpReOpenBuf,   //オープンしたファイルに関する情報を受
                                  け取るOFSTRUCT構造体へのポインタ
    WORD         wStyle         //実行する動作
);
```

V　B

Declare Function LZOpenFile Lib "lz32.dll" Alias "LZOpenFileA" (ByVal lpszFile As String, lpOf As OFSTRUCT, ByVal style As Long) As Long

パラメータ

- *lpFileName*　オープンするファイル名が格納された文字列へのポインタ。
- *lpReOpenBuf*　ファイルをオープンするときにファイルに関する情報を格納するためのOFSTRUCT構造体へのポインタ。
- *wStyle*　LZOpenFile関数の実行にともなう動作を次の値の組み合わせで指定する。各スタイルはビットごとのOR演算子を利用して複数指定が可能。

値	意味
OF_CANCEL	Win32アプリケーションでは無視。キャンセルボタンはOF_PROMPTスタイルでサポートされる
OF_CREATE	新規ファイルを作成。すでにファイルが存在する場合にはファイル長が0に切りつめられる
OF_DELETE	ファイルを削除
OF_EXIST	ファイルをオープン後クローズする。ファイルの存在テストに利用する
OF_PARSE	OFSTRUCT構造体へ情報を格納する
OF_PROMPT	指定されたファイルが見つからなかった場合にユーザーにダイアログボックスを表示する。キャンセルボタンが選択された場合にはLZOpenFile関数は"file not found"エラーメッセージを返す
OF_READ	読み取り専用属性でファイルをオープンする
OF_READWRITE	読み書き可能な属性でファイルをオープンする
OF_REOPEN	再オープン用のバッファを利用してファイルをオープンする
OF_SHARE_DENY_NONE	他のプログラムから読み書きのアクセスが可能な形式でファイルをオープンする。ファイルがほかのプログラムによってすでに互換属性でオープンされている場合にはLZOpenFileの実行は失敗する

OF_SHARE_DENY_READ	他のプログラムから読み取りアクセスが不可能な形式でファイルをオープンする。ファイルがほかのプログラムによってすでに互換属性もしくは読み取り属性でオープンされている場合にはLZOpenFileの実行は失敗する
OF_SHARE_DENY_WRITE	他のプログラムから書き込みアクセスが不可能な形式でファイルをオープンする。ファイルがほかのプログラムによってすでに互換属性もしくは書き込み属性でオープンされている場合にはLZOpenFileの実行は失敗する
OF_SHARE_EXCLUSIVE	排他属性でファイルをオープンする。他のプログラムからはファイルアクセスはできない。自分自身を含む他のプログラムからこの属性でファイルがオープンされている場合にはLZOpnFile関数の実行は失敗する
OF_WRITE	書き込み専用属性でファイルをオープンする

戻り値

正常終了	*wStyle*パラメータがOF_READ以外の場合にはファイルハンドル。*wStyle*パラメータにOF_READを設定して圧縮ファイルをオープンした場合には、特殊なファイルハンドルを返す。

異常終了時には、次の戻り値のいずれかが返される。

LZERROR_BADINHANDLE	*lpFileName*パラメータに無効なファイルハンドルが指定された。ファイルが見つからない
LZERROR_GLOBALLOC	上限数を上回るファイルを既にオープンしている。もしくはローカルメモリの取得に失敗した

解　説

*wStyle*パラメータがOF_READ（もしくはOF_READとOF_SHARE_*の組み合わせ）でファイルが圧縮されている場合には、LZOpenFile関数はLZInitを呼び出す。LZInitはファイルの展開操作に必要な初期化作業を実行する。

参　照

→LZClose,LZInit,LZRead

Error

Beep	kernel32.dll	95	NT
ビープ音を発生			

V C

```
BOOL Beep(
    DWORD   dwFreq,       //音の周波数をヘルツ（Hz）で指定
    DWORD   dwDuration    //音の再生時間をミリ秒（ms）で指定
    );
```

V B

```
Declare Function Beep Lib "kernel32" Alias "Beep" (ByVal dwFreq As Long, ByVal dwDuration As Long) As Long
```

パラメータ

・*dwFreq*　プラットフォームによって以下の通り指定を行う。

Windows 95	無視される。
WindowsNT	周波数をヘルツ（Hz）で指定する。指定できる範囲は37（0x25）から32,767（0x7FFF）である。

・*dwDuration*　プラットフォームによって以下の通り指定を行う。

Windows 95	無視される。
WindowsNT	再生時間をミリ秒（ms）で指定する。パラメータに-1を指定した場合には再びBeep関数が呼び出されるまでBeep音が再生される。

戻り値

正常終了	TRUE
異常終了	FALSE

拡張エラー情報はGetLastError関数で取得。

解　説

Windows 95の場合には、Beep関数のすべてのパラメータは無視される。コンピュータにサウンドカードがインストールされている場合には、デフォルトのサウンドが再生される。サウンドカードがインストールされていない場合には、標準のシステムBeep音が鳴る。
WindowsNTの場合、Beep関数は再生が終了するまでアプリケーションに制御を返さない。ただし、*dwDuration*パラメータに-1が指定された場合には、関数の実行後ただちに制御が戻される。

参　照

MapWindowPoints,POINT,ScreenToClient

ExitWindows	user32.dll	95	NT
Windowsをシャットダウン			

V C

```
BOOL ExitWindows(
    DWORD   dwReserved,     //予約済み0
    UINT    uReserved       //予約済み0
    );
```

V B

Declare Function ExitWindows Lib "user32" Alias "ExitWindows" (ByVal dwReserved As Long, ByVal uReturnCode As Long) As Long

パラメータ

- *dwReserved*　予約済みで0でなければならない。
- *uReserved*　予約済みで0でなければならない。

戻り値

正常終了	TRUE
異常終了	FALSE

拡張エラー情報はGetLastError関数で取得。

解　説

Windows 95の場合、ExitWindows関数は呼び出し元のアプリケーションを除くすべての実行中アプリケーションに対してWM_QUERYENDSESSIONメッセージを送信する。すべての実行中アプリケーションからシャットダウンを許可するTRUEの戻り値が返された場合は、シャットダウンが実行される。いずれかのアプリケーションからシャットダウンを拒否するFALSEの戻り値があった場合には、シャットダウン処理はキャンセルされる。

WindowsNTでは、ExitWindows関数はすべての実行中アプリケーションに対してWM_QUERYENDSESSIONメッセージを送信してシャットダウン処理の実行を確認する。

シャットダウン処理中にWindowsは新しいアプリケーションを起動することはできない。

参　照

DefWindowProc,ExitWindowsEx

GetLastError	kernel32.dll	95	NT
拡張エラーコードを返す			

V C　DWORD GetLastError();

V B　Declare Function GetLastError Lib "kernel32" Alias "GetLastError" () As Long

戻り値　最後に呼び出されたAPI関数の処理中に関数が実行したSetLastError関数の引数が最終エラーコードとして戻り値になる。

解説　戻り値となるエラーコードは32ビット値である(31ビット目が非常に重要な意味を表わす)。29ビット目はアプリケーション予約のエラーコードビットである。アプリケーションでユーザー定義のエラーコードを定義する場合には、このビットを利用する。これによって、Windowsが返すいかなるエラーコードともバッティングすることなくエラーコードの設定が可能になる。

参照　SetLastError,SetLastErrorEx

File

CopyFile	kernel32.dll	95	NT
ファイルをコピー			

V C

```
BOOL CopyFile(
        LPCTSTR  lpExistingFileName,  //コピー元ファイル名を格納した文字列へのポインタ
        LPCTSTR  lpNewFileName,       //コピー先ファイル名の文字列を格納した文字列へのポインタ
        BOOL     bFailIfExists        //ファイル存在時の動作
        );
```

V B

Declare Function CopyFile Lib "kernel32" Alias "CopyFileA" (ByVal lpExistingFileName As String, ByVal lpNewFileName As String, ByVal bFailIfExists As Long) As Long

パラメータ

- *lpExistingFileName*　コピー元ファイル名を格納した文字列へのポインタ。
- *lpNewFileName*　コピー先ファイル名を格納した文字列へのポインタ。
- *bFailIfExists*　*lpNewFileName*パラメータで指定されるファイルが、コピー先にすでに同名のファイルとして存在していた場合の操作を指定する。次の値のいずれかを指定する。

値	意味
TRUE	関数は異常終了し、ファイルのコピーは実行されない
FALSE	関数は正常終了し、ファイルは上書きコピーされる

戻り値

正常終了	TRUE
異常終了	FALSE

拡張エラー情報はGetLastError関数で取得。

参照

CreateFile,→MoveFile

GetCurrentDirectory	kernel32.dll	95	NT
プロセスの現在のディレクトリを返す			

V C

```
DWORD GetCurrentDirectory(
    DWORD   nBufferLength,    //ディレクトリ名を格納するバッファサイズ
    LPTSTR  lpBuffer          //ディレクトリ名を格納するバッファへのポインタ
    );
```

V B

```
Declare Function GetCurrentDirectory Lib "kernel32" Alias "GetCurrentDirectoryA"
(ByVal nBufferLength As Long, ByVal lpBuffer As String) As Long
```

パラメータ

- *nBufferLength*　カレントディレクトリ名を格納するためのバッファサイズを指定する。バッファサイズはディレクトリ名の文字列の長さと終端のNULL文字の合計。
- *lpBuffer*　カレントディレクトリ名を格納するためのNULLで終わる文字列バッファへのポインタ。

戻り値

正常終了	終端のNULL文字を除いたカレントディレクトリ名の長さ
異常終了	0

拡張エラー情報はGetLastError関数で取得。

参照

CreateDirectory,GetSystemDirectory,→GetWindowsDirectory,RemoveDirectory,
SetCurrentDirectory

GetDiskFreeSpace	kernel32.dll	95	NT
空きディスク容量の総量を返す			

V C

```
BOOL GetDiskFreeSpace(
    LPCTSTR   lpRootPathName,          //ルートディレクトリ名を格納した文字列へのポインタ
    LPDWORD   lpSectorsPerCluster,     //クラスタあたりのセクタ数を格納するバッファへのポインタ
    LPDWORD   lpBytesPerSector,        //セクタあたりのバイト数を格納するバッファへのポインタ
    LPDWORD   lpNumberOfFreeClusters,  //空きクラスタの数を格納するバッファへのポインタ
    LPDWORD   lpTotalNumberOfClusters  //総クラスタ数を格納するバッファへのポインタ
);
```

V B

Declare Function GetDiskFreeSpace Lib "kernel32" Alias "GetDiskFreeSpaceA" (ByVal lpRootPathName As String, lpSectorsPerCluster As Long, lpBytesPerSector As Long, lpNumberOfFreeClusters As Long, lpTtoalNumberOfClusters As Long) As Long

パラメータ

- *lpRootPathName*　NULLで終わるルートディレクトリ名を格納した文字列へのポインタ。このパラメータにNULLを指定した場合は、カレントディレクトリのルートディレクトリが指定されるのと同一である。
- *lpSectorsPerCluster*　クラスタあたりのセクタ数を格納するバッファへのポインタ。
- *lpBytesPerSector*　セクタあたりのバイト数を格納するバッファへのポインタ。
- *lpNumberOfFreeClusters*　空きクラスタの数を格納するバッファへのポインタ。
- lpTotalNumberOfClusters　総クラスタ数を格納するバッファへのポインタ。

戻り値

正常終了	TRUE
異常終了	FALSE

拡張エラー情報はGetLastError関数で取得。

参照

GetDriveType

GetFileAttributes	kernel32.dll	95	NT
ファイル属性を返す			

V C

```
DWORD GetFileAttributes(
    LPCTSTR    lpFileName    //ファイルもしくはディレクトリ名を格納した文字列へのポインタ
);
```

V B

```
Declare Function GetFileAttributes Lib "kernel32" Alias "GetFileAttributesA" (ByVal lpFileName As String) As Long
```

パラメータ

・*lpFileName*　ファイル名もしくはディレクトリ名を格納したNULLで終わる文字列へのポインタ。

戻り値

正常終了	属性を示す次の値
異常終了	0xFFFFFFFF

拡張エラー情報はGetLastError関数で取得。

値	意味
FILE_ATTRIBUTE_ARCHIVE	アーカイブ属性
FILE_ATTRIBUTE_COMPRESSED	圧縮属性。ファイルの場合にはデータ全体が圧縮されていることを、ディレクトリの場合には新規作成されたばかりのサブディレクトリであることを示している。
FILE_ATTRIBUTE_DIRECTORY	ディレクトリ
FILE_ATTRIBUTE_HIDDEN	非表示属性
FILE_ATTRIBUTE_NORMAL	通常ファイル
FILE_ATTRIBUTE_READONLY	読み込み専用
FILE_ATTRIBUTE_SYSTEM	システムファイル

参　照

DeviceIOControl,FindFirstFile,FindNextFile,SetFileAttributes

GetFileSize	kernel32.dll	95	NT
ファイルサイズの取得			

V C

```
DWORD GetFileSize(
    HANDLE      hFile,           //サイズを取得するファイルのハンドル
    LPDWORD     lpFileSizeHigh,  //ファイルサイズの上位ワードを格納する変
                                   数へのポインタ
    );
```

V B

Declare Function GetFileSize Lib "kernel32" Alias "GetFileSize" (ByVal hFile As Long, lpFileSizeHigh As Long) As Long

パラメータ

- *hFile* サイズが戻されるファイルのハンドルを指定する。ハンドルはGENERIC_READまたはGENERIC_WRITEアクセスによって作成されたものでなければならない。
- *lpFileSizeHigh* ファイルサイズの上位ワードが戻される変数へのポインタ。上位ワードを必要としない場合には、このパラメータはNULLでである。

戻り値

正常終了	ファイルサイズの下位ワード。*lpFileSizeHigh*がNULLでない場合には、ファイルサイズの上位ワードを、パラメータで示された変数に入れる。
異常終了（*lpFileSizeHigh* = NULL）	0xFFFFFFFF。拡張エラー情報はGetLastError関数で取得する。
異常終了（*lpFileSizeHigh* = NULL以外）	0xFFFFFFFF。GetLastError関数はNO_ERRORを返す。

解　説

*hFile*のためのファイルの種類を決定するためには、GetFileType関数を使う。GetFileSize関数は、ファイルの未圧縮時のサイズを取得する。ファイルの圧縮されたサイズを取得するには、GetCompressedFileSize関数を利用する。

参　照

GetCompressedFileSize,→GetFileType

GetFileType	kernel32.dll	95	NT
ファイル種類を取得			

V　C

```
DWORD GetFileType(
    HANDLE    hFile    //ファイルのハンドル
    );
```

V　B

Declare Function GetFileType Lib "kernel32" Alias "GetFileType" (ByVal hFile As Long) As Long

パラメータ

・*hFile*　　ファイル種類を判別するためのオープンされたファイルのハンドル

戻り値

戻り値は次の値のいずれかになる。

値	意味
FILE_TYPE_UNKNOWN	不明
FILE_TYPE_DISK	ディスクファイル
FILE_TYPE_CHAR	プリンタもしくはコンソール文字
FILE_TYPE_PIPE	名前付きもしくは名前なしのパイプ

参　照

→GetFileSize,→GetFileTime

GetFullPathName	kernel32.dll	95	NT
ファイル名のパス名を取得			

V C

```
DWORD GetFullPathName(
    LPCTSTR  lpFileName,     //パス名を取得したいファイル名を格納した文字列
                               へのポインタ
    DWORD    nBufferLength, //パス名を格納するバッファサイズ
    LPTSTR   lpBuffer,       //パス名を格納するバッファへのポインタ
    LPTSTR   *lpFilePart     //ファイル名を格納するバッファへのポインタ
);
```

V B

Declare Function GetFullPathName Lib "kernel32" Alias "GetFullPathNameA" (ByVal lpFileName As String, ByVal nBufferLength As Long, ByVal lpBuffer As String, ByVal lpFilePart As String) As Long

パラメータ

- *lpFileName*　パス名を取得したいファイル名を格納したNULLで終わる文字列へのポインタ。ファイル名はMS-DOSの8.3形式でも長いファイル名の形式でも指定が可能。
- *nBufferLength*　ドライブ名から始まるパス名を格納するバッファのサイズ
- *lpBuffer*　NULLで終わるパス名を格納するバッファへのポインタを指定する。
- *lpFilePart*　*lpBuffer*パラメータに返されるフルパス名の文字列中に含まれるファイル名。

戻り値

正常終了	*lpBuffer*パラメータに返された文字列の長さ。ただし終端のNULL文字は含まれていない
異常終了	0

拡張エラー情報はGetLastError関数で取得。

解　説

GetFullPathName関数は、指定されたファイル名にカレントドライブのドライブ名とカレントディレクトリ名を結合して、フルパス形式のファイル名を生成する。関数の実行によって返されるフルパス名についての整合性、妥当性検査は行われない。

参　照

GetShortPathName,GetTempPath,SearchPath

LockFile	kernel32.dll	95	NT
ファイルをロック			

V　C

```
BOOL LockFile(
    HANDLE  hFile,                        //ロックするファイルのハンドル
    DWORD   dwFileOffsetLow,              //ロック領域の下位ワード
    DWORD   dwFileOffsetHigh,             //ロック領域の上位ワード
    DWORD   nNumberOfBytesToLockLow,      //ロック領域の長さの下位ワード
    DWORD   nNumberOfBytesToLockHigh      //ロック領域の長さの上位ワード
    );
```

V　B

```
Declare Function LockFile Lib "kernel32" Alias "LockFile" (ByVal hFile As Long, ByVal dwFileOffsetLow As Long, ByVal dwFileOffsetHigh As Long, ByVal nNumberOfBytesToLockLow As Long, ByVal nNumberOfBytesToLockHigh As Long) As Long
```

パラメータ

- *hFile*　GENERIC_READまたはGENERIC_WRITEで作成されたロックするファイルを識別するハンドル。
- *dwFileOffsetLow*　ロックする領域の開始位置(ファイルの先頭からのオフセット値)の下位ワードを指定する。
- *dwFileOffsetHigh*　ロックする領域の開始位置(ファイルの先頭からのオフセット値)の上位ワードを指定する。
- *nNumberOfBytesToLockLow*　ロックする領域の長さの下位ワードを指定する。
- *nNumberOfBytesToLockHigh*　ロックする領域の長さの上位ワードを指定する。

戻り値

正常終了	TRUE
異常終了	FALSE

拡張エラー情報はGetLastError関数で取得する。

解　説

ファイル領域をロックすることで、指定された領域の占有アクセス権が得られる。またファイルの終わりを越える領域をロックしてもエラーとならない。

参　照

CreateFile,UnlockFile

MoveFile	kernel32.dll	95	NT
ファイル名を変更			

V　C

```
BOOL MoveFile(
    LPCTSTR   lpExistingFileName, //移動するファイル名が格納された文字列へのポインタ
    LPCTSTR   lpNewFileName       //移動先の情報が格納された文字列へのポインタ
);
```

V　B

Declare Function MoveFile Lib "kernel32" Alias "MoveFileA" (ByVal lpExisting FileName As String, ByVal lpNewFileName As String) As Long

パラメータ

・*lpExistingFileName*　移動するファイル名が格納されたNULLで終わる文字列へのポインタを指定する。

・*lpNewFileName*　移動先のファイル名やディレクトリ名を格納したNULLで終わる文字列へのポインタを指定する。ここで指定するファイル名やディレクトリ名がすでに存在してはならない。

戻り値

正常終了	TRUE
異常終了	FALSE

拡張エラー情報はGetLastError関数で取得。

解　説

MoveFile関数は、ファイルもしくはディレクトリとその下層に位置するファイルやサブディレクトリすべてを別の場所へ移動する。

参　照

→CopyFile,MoveFileEx

ReadFile	kernel32.dll	95	NT
ファイルから読み取る			

V C

```
BOOL ReadFile(
    HANDLE        hFile,                  //ファイルのハンドル
    LPVOID        lpBuffer,               //読み取ったデータを格納する
                                            バッファへのポインタ
    DWORD         nNumberOfBytesToRead,   //読み取るサイズ
    LPDWORD       lpNumberOfBytesRead,    //読み取りを開始する位置への
                                            ポインタ
    LPOVERLAPPED  lpOverlapped            //OVERLAPPED構造体へのポイ
                                            ンタ
    );
```

V B

Declare Function ReadFile Lib "kernel32" Alias "ReadFile" (ByVal hFile As Long, lpBuffer As Any, ByVal nNumberOfBytesToRead As Long, lpNumberOfBytesRead As Long, lpOverlapped As OVERLAPPED) As Long

パラメータ

- *hFile*　読み取りを開始するファイルのハンドルを指定する。
- *lpBuffer*　ファイルから読み取った情報を格納するためのバッファへのポインタを指定する。
- *nNumberOfBytesToRead*　ファイルから読み取るデータのサイズを指定する。
- *lpNumberOfBytesRead*　ファイルから読み取りを開始するバイト位置を示すポインタを指定する。
- *lpOverlapped*　OVERLAPPED構造体へのポインタ。*hFile*がFILE_FLAG_OVERLAPPEDによって作成されている場合には、この構造体が必要である。

戻り値

正常終了	TRUE
異常終了	FALSE

拡張エラー情報はGetLastError関数で取得する。

解　説

ファイルの一部が別のプロセスによってロックされ、ロックされた部分と読み出し要求部分が重複している場合、関数は失敗する。

参　照

CreateFile,GetCommTimeouts,GetOverlappedResult,→OVERLAPPED,
→PeekNamedPipe,ReadFileEx,SetCommTimeouts,WriteFile

RemoveDirectory	kernel32.dll	95	NT
既存のディレクトリを削除			

V C

```
BOOL RemoveDirectory(
    LPCTSTR  lpPathName    //ディレクトリを識別するパス名へのポインタ
    );
```

V B

Declare Function RemoveDirectory Lib "kernel32" Alias "RemoveDirectoryA" (ByVal lpPathName As String) As Long

パラメータ

・*lpPathName*　削除するディレクトリを識別するNULLで終わる文字列へのポインタを指定する。

戻り値

正常終了	TRUE
異常終了	FALSE

拡張エラー情報はGetLastError関数で取得。

参　照

CreateDirectory

Help

GetMenuContextHelpId	user32.dll	95	
メニューコンテキストヘルプIDを取得			

VC

```
DWORD GetMenuContextHelpId(
    HMENU   hmenu    //ヘルプコンテキストIDを取得するメニューのハンドル
    );
```

VB

Declare Function GetMenuContextHelpId Lib "user32" Alias "GetMenuContextHelpId" (ByVal hMenu As Long) As Long

パラメータ　・*hmenu*　ヘルプコンテキストIDを取得するメニューのハンドルを指定する。

戻り値　メニュー項目が存在する場合にはコンテキストヘルプIDを返す。

GetWindowContextHelpId	user32.dll	95	
ウィンドウコンテキストヘルプIDを取得			

VC

```
DWORD GetWindowContextHelpId(
    HWND hwnd    //ヘルプコンテキストIDを取得するウィンドウのハンドル
    );
```

VB

Declare Function GetWindowContextHelpId Lib "user32" Alias "GetWindowContextHelpId" (ByVal hWnd As Long) As Long

パラメータ　・*hmenu*　ヘルプコンテキストIDを取得するウィンドウのハンドルを指定する。

戻り値　ウィンドウが存在する場合にはコンテキストヘルプIDを返す。

WinHelp	user32.dll	95	NT
Windowsヘルプアプリケーションを起動し、コンテキスト情報やトピック情報を渡す			

VC

```
BOOL WinHelp(
    HWND     hWndMain,   //ヘルプを要求するウィンドウを識別するハンドル
    LPCTSTR  lpszHelp,   //ヘルプに表示させるヘルプファイル名を格納する
                          NULLで終わる文字列へのポインタ
    UINT     uCommand,   //要求するヘルプの種類
    DWORD    dwData      //追加データ
    );
```

V B　Declare Function WinHelp Lib "user32" Alias "WinHelpA" (ByVal hwnd As Long, ByVal lpHelpFile As String, ByVal wCommand As Long, ByVal dwData As Long) As Long

パラメータ

・*hWndMain*　ヘルプを要求したウィンドウを識別するハンドル。WinHelp関数はこのパラメータを利用してアプリケーションを管理する。

・*lpszHelp*　ヘルプに表示させるヘルプファイルのファイル名とディレクトリパスを格納したNULLで終わる文字列へのポインタを指定する。ヘルプトピックを1次ウィンドウではなく2次ウィンドウに表示させる場合には、ファイル名の後に不等号 (>) と2次ウィンドウの名前を指定する。2次ウィンドウの名前は、ヘルププロジェクトファイル（.HPJ）の[WINDOWS]セクションであらかじめ定義しておかなければならない。

・*uCommand*　要求するヘルプの種類を指定する。指定できる値の一覧と*dwData*パラメータの値の意味については解説を参照。

・*dwData*　追加データを指定する。利用可能な値は*uCommand*パラメータの値により異なる。

戻り値

正常終了	TRUE
異常終了	FALSE

メニュー項目が存在する場合にはコンテキストヘルプIDを返す。

解　説　要求したヘルプのウィンドウをクローズする前にアプリケーションは*uCommand*にHELP_QUITを指定してWinHelpを呼び出さなければならない。すべてのアプリケーションがこの動作を実行するまでWindowsヘルプは終了しない。

uCommand	動作	*dwData*
HELP_COMMAND	ヘルプマクロを実行	実行するヘルプマクロを表す文字列へのポインタ
HELP_CONTENTS	.HPJファイルの[OPTIONS]セクション内のCONTENTSオプションで定義されたヘルプの目次トピックを表示	無視される。0に設定
HELP_CONTEXT	.HPJファイルの[MAP]セクションで定義されたコンテキスト番号が識別する特定のヘルプトピックを表示	トピックのコンテキスト番号を表す符号なし整数
HELP_CONTEXTPOPUP	.HPJファイルの[MAP]セクションで定義されたコンテキスト番号が識別する特定のヘルプトピックをポップアップウィンドウに表示	トピックのコンテキスト番号を表す符号なし整数

HELP_FORCEFILE	WinHelp関数が正しいヘルプファイルを表示していることを保証する。現在正しいヘルプファイルが表示されている場合には何も実行されない。間違ったヘルプファイルが表示されている場合にはWinHelp関数は正しいファイルをオープン	無視される。0に設定
HELP_HELPONHELP	［ヘルプの使い方］ファイルを示す目次トピックを表示	無視される。0に設定
HELP_INDEX	ヘルプトピックの検索ダイアログに目次を表示	無視される。0に設定
HELP_KEY	*dwData*パラメータに渡されるキーワードと正確に一致するキーワードがリスト内に1つある場合にそのトピックを表示	トピックのキーワードを表す文字列へのポインタ
HELP_MULTIKEY	代替キーワードテーブル内のキーワードが識別するヘルプトピックを表示	MULTIKEYHELP構造体へのポインタ
HELP_PARTIALKEY	*dwData*パラメータに渡されるキーワードと正確に一致するキーワードがリスト内に1つある場合にはそのトピックを表示。一致するキーワードが複数ある場合には［トピックの検索］ダイアログボックスを表示して［ジャンプ］リストボックスの中にトピックをリスト表示。一致するキーワードがない場合には［トピックで検索］ダイアログボックスを表示	トピックのキーワードを表す文字列へのポインタ
HELP_QUIT	ヘルプが不要になったことをWindowsヘルプに知らせる。ほかのアプリケーションがヘルプを要求していない場合にはWindowsヘルプがクローズ	無視される。0に設定
HELP_SETCONTENTS	ユーザーがF1キーを押したときにヘルプに表示させる目次トピックを決定	アプリケーションで目次トピックとして表示させたいトピックのコンテキスト番号を表す符号なし整数

HELP_SETINDEX	指定されたトピックを目次に設定表示する	目次トピックを表す文字列へのポインタ
HELP_SETWINPOS	ヘルプウィンドウがアイコン化されていたりメモリ内にある場合にウィンドウを表示して渡されたデータに基づいてウィンドウの位置を調整する	HELPWININFO構造体へのポインタ

参照 →HELPWININFO,→MULTIKEYHELP

Memory Management

GlobalAlloc

GlobalAlloc	kernel32.dll	95	NT

グローバルヒープからメモリを割り当てる

V C

```
HGLOBAL GlobalAlloc(
    UINT    uFlags,    //メモリを割り当てる方法
    DWORD   dwBytes    //割り当てられるバイト数
    );
```

V B

Declare Function GlobalAlloc Lib "kernel32" Alias "GlobalAlloc" (ByVal wFlags As Long, ByVal dwBytes As Long) As Long

パラメータ

・*uFlags*　　メモリを割り当てる方法を次の値で指定する。

値	意味
GMEM_FIXED	固定メモリを割り当てる。GMEM_FIXEDとGMEM_MOVEABLEは同時には指定できない
GMEM_MOVEABLE	移動可能メモリを割り当てる。GMEM_FIXEDとGMEM_MOVEABLEは同時に指定できない
GPTR	GMEM_FIXEDとGMEM_ZEROINITの組み合わせ
GHND	GMEM_MOVEABLEとGMEM_ZEROINITの組み合わせ
GMEM_DDESHARE	共有可能メモリを割り当てる。ダイナミックデータ交換（DDE）だけに使われ、GEM_SHAREと等価
GMEM_DISCARDABLE	廃棄可能メモリを割り当てる。必ずGMEM_MOVEABLEとともに使用する
GMEM_LOWER	無視される。Windows 3. xとの互換性のために残されている
GMEM_NOCOMPACT	割り当て要求に対してメモリのコンパクションや廃棄を行わない
GMEM_NODISCARD	割り当て要求に対してメモリの廃棄を行わない

GMEM_NOT_BANKED	無視される。Windows 3. xとの互換性のために残されている
GMEM_NOTIFY	無視される。Windows 3. xとの互換性のために残されている
GMEM_SHARE	ほかのアプリケーションと共有できるメモリを割り当てる。GMEM_DDESHAREと等価
GMEM_ZEROINIT	メモリ内容を0に初期化

・*dwBytes*　割り当てるメモリのバイト数を指定する。

戻り値

正常終了	割り当てられたメモリオブジェクトのハンドル
異常終了	NULL

拡張エラー情報はGetLastError関数で取得。

解　説

アプリケーションでGlobalAlloc関数により返されたハンドルをポインタに変換するには、GlobalLock関数を使わなければならない。
関数が正常に終了した場合は、少なくとも要求された最大量を割り当てる。割り当てられたメモリの最大量が要求された量よりも多くなった場合、アプリケーションはそのメモリをすべて使用できる。グローバルメモリオブジェクトのサイズを決めるには、アプリケーションでGlobalSize関数を使用する。
グローバルメモリオブジェクトを解除するには、GlobalFree関数を使わなければならない。割り当てられたメモリオブジェクトのサイズや属性を変更する場合には、GlobalReAlloc関数を使うことで実現できる。

参　照

GlobalFree,GlobalLock,GlobalReAlloc,GlobalSize,→LocalAlloc

HeapAlloc　kernel32.dll　95　NT

ヒープからメモリを割り当てる

V　C

```
LPVOID HeapAlloc(
    HANDLE  hHeap,    //ヒープブロックを識別するハンドル
    DWORD   dwFlags,  //メモリの割り当て方法
    DWORD   dwBytes   //割り当てるメモリのサイズ
    );
```

V　B

```
Declare Function HeapAlloc Lib "kernel32" Alias "HeapAlloc" (ByVal hHeap As Long, ByVal dwFlags As Long, ByVal dwBytes As Long) As Long
```

パラメータ

・*hHeap*　メモリが割り当てられるヒープブロックを識別するハンドルを指定する。パラメータは、HeapCreateまたはGetProcessHeap関数で戻されたハンドルである。

・*dwFlags*　メモリを割り当てる方法を指定する。次の値のいずれかを指定する。

値	意味
HEAP_GENERATE_EXCEPTIONS	関数が失敗した場合にOSが例外を返す。
HEAP_NO_SERIALIZE	関数がヒープブロックにアクセスする際に排他的な

	アクセスを実行しない。
HEAP_ZERO_MEMORY	割り当てられるメモリは0で初期化される。

・*dwBytes*　割り当てられるメモリのサイズを指定する。

戻り値

正常終了	割り当てられたメモリブロックへのポインタ
異常終了	NULL

HEAP_GENERATE_EXCEPTIONSフラグが指定され、関数が異常終了した場合には、次の例外が返される。

STATUS_NO_MEMORY	要求されたサイズのメモリを割り当てることができない。
STATUS_ACCESS_VIOLATION	指定されたパラメータが正しくない。

解説

HeapAlloc関数が正常に終了した場合、少なくとも要求されたメモリのサイズが割り当てられる。割り当てられたブロックの実際のサイズを確認するためには、HeapSize関数を使う。
割り当てられたブロックをメモリから解放するためには、HeapFree関数を使う。
HeapAllocにより割り当てられたメモリは固定である。

参照

GetProcessHeap,HeapCreate,HeapDestroy,HeapFree,HeapReAlloc,HeapSize,
→SetLastError

LocalAlloc　kernel32.dll　95　NT

ローカルヒープからメモリを割り当てる

VC

```
HLOCAL LocalAlloc(
    UINT    uFlags,    //メモリを割り当てる方法
    UINT    uBytes     //割り当てるメモリのバイト数
    );
```

VB

```
Declare Function LocalAlloc Lib "kernel32" Alias "LocalAlloc" (ByVal wFlags As Long, ByVal wBytes As Long) As Long
```

パラメータ

・uFlags　次の値の組み合わせからメモリを割り当てる方法を指定する。

値	意味
LMEM_FIXED	固定メモリを割り当てる。LMEM_MOVEABLEとともに指定はできない
LMEM_MOVEABLE	移動可能なメモリを割り当てる。LMEM_FIXEDとともに指定はできない
LPTR	LMEM_FIXEDとLMEM_ZEROINITの組み合わせ
LHND	LMEM_MOVEABLEとLMEM_ZEROINITの組み合わせ
NONZEROLHND	LMEM_MOVEABLEと等価

NONZEROLPTR	LMEM_FIXEDと等価
LMEM_DISCARDABLE	廃棄可能なメモリを割り当てる。LMEM_FIXEDとともに指定はできない。Win32アプリケーションではこの指定は無視される
LMEM_NOCOMPACT	割り当て要求を満たすためのメモリのコンパクションや廃棄を行わない
LMEM_NODISCARD	割り当て要求を満たすためのメモリの廃棄を行わない
LMEM_ZEROINIT	メモリ内容を0で初期化する

・*uBytes*　割り当てるメモリのバイト数を指定する。

戻り値

正常終了	割り当てられたメモリオブジェクトのハンドル
異常終了	NULL

拡張エラー情報はGetLastError関数で取得。

解　説

LocalAlloc関数は移動可能なメモリを割り当てると、メモリのローカルハンドルを返す。アプリケーションは、メモリにアクセスするためにLocalLock関数を使用してハンドルをポインタに変換しなければならない。
LocalAlloc関数は固定メモリを割り当てると、メモリを指すポインタを返す。アプリケーションは戻り値をポインタにキャストするだけでメモリにアクセスできる。

参　照

→GlobalAlloc,LocalFree,LocalLock,LocalReAlloc,LocalSize

VirtualAlloc

kernel32.dll　|　95　|　NT

仮想メモリを割り当てる

V　C

```
LPVOID VirtualAlloc(
    LPVOID  lpAddress,         //確保する領域へのポインタ
    DWORD   dwSize,            //領域のサイズ
    DWORD   flAllocationType,  //割り当ての方法
    DWORD   flProtect          //アクセス権
    );
```

V　B

```
Declare Function VirtualAlloc Lib "kernel32" Alias "VirtualAlloc" (lpAddress As Any, ByVal dwSize As Long, ByVal flAllocationType As Long, ByVal flProtect As Long) As Long
```

パラメータ

・*lpAddress*　割り当てのために、要求する領域のアドレスを指定する。ホストのコンピュータのページサイズを決定するためには、GetSystemInfo関数を使う。

・*dwSize*　領域のサイズを指定する。

・*flAllocationType*　領域を割り当てるときの方法を指定する。次の値のいずれかを指定する。

値	意味
MEM_COMMIT	ページの指定された領域ディスクにおいてメモリまたはページングファイルの物理的な領域を割り当てる。
MEM_RESERVE	プロセスがあらゆる物理的な領域を確保する。確保された範囲は、それが解放されるまで、他の割り当てオペレーション（malloc関数、LocalAlloc関数など）によって利用することはできない。
MEM_TOP_DOWN	可能な限り上位のメモリを割り当てる。

・*flProtect*　アクセス保護のタイプを指定する。次のフラグのいずれかを指定する。

値	意味
PAGE_READONLY	読み取り専用
PAGE_READWRITE	読み書き
PAGE_EXECUTE	実行可能
PAGE_EXECUTE_READ	実行と読み取り
PAGE_EXECUTE_READWRITE	実行と読み書き
PAGE_GUARD	保護
PAGE_NOCACHE	キャッシュ無効

戻り値

正常終了	割り当てられたページアドレス
異常終了	NULL

拡張エラー情報はGetLastError関数で取得する。

解説

VirtualAlloc関数は次の操作を実行できる。

・前の呼び出しでVirtualAlloc関数に確保されたページ領域を確保する
・あいているページ領域を確保する
・上記の両方

参照

→GlobalAlloc,→HeapAlloc,VirtualFree,VirtualLock,VirtualProtect,VirtualQuery

ZeroMemory　——　95　NT

メモリブロックの内容をすべて0にする

VC

```
void ZeroMemory(
    PVOID   Destination,    //0で初期化するメモリブロックのアドレス
    DWORD   Length          //0で初期化するメモリブロックのバイト数
);
```

パラメータ

・*Destination*　0で初期化するメモリブロックのアドレスを指定する。
・*Length*　0で初期化するメモリブロックのバイト数を指定する。

参照

CopyMemory,FillMemory,MoveMemory

Pipe

ConnectNamedPipe	kernel32.dll	95	NT
名前付きパイプに接続			

V C

```
BOOL ConnectNamedPipe(
    HANDLE          hNamedPipe,    //名前付きパイプを識別するハンドル
    LPOVERLAPPED    lpOverlapped   //OVERLAPPED構造体へのポインタ
    );
```

V B

Declare Function ConnectNamedPipe Lib "kernel32" Alias "ConnectNamedPipe" (ByVal hNamedPipe As Long, lpOverlapped As OVERLAPPED) As Long

パラメータ

・*hNamedPipe*　名前付きパイプの終端を識別するハンドルを指定する。このハンドルはCreateNamedPipe関数により戻される。

・*lpOverlapped*　OVERLAPPED構造体へのポインタを指定する。

戻り値

正常終了	TRUE
異常終了	FALSE

拡張エラー情報はGetLastError関数で取得する。

解　説

ConnectNamedPipeを使うことで、名前付きパイプのサーバープロセスは、以前の別のクライアントプロセスとの接続を利用して、新しく作成されたパイプでも利用できる。

参　照

CallNamedPipe,CreateEvent,CreateFile,→CreateNamedPipe,
→DisconnectNamedPipe,GetOverlappedResult,SetNamedPipeHandleState,
SleepEx,WaitForMultipleObjects,WaitForMultipleObjectsEx,WaitForSingleObject,
WaitForSingleObjectEx,→OVERLAPPED

CreateNamedPipe	kernel32.dll		NT
名前付きパイプを作成			

V C

```
HANDLE CreateNamedPipe(
    LPCTSTR    lpName,            //名前付きパイプへのポイ
                                    ンタ
    DWORD      dwOpenMode,        //パイプのオープンモード
    DWORD      dwPipeMode,        //パイプモード
    DWORD      nMaxInstances,     //インスタンスの最大値
    DWORD      nOutBufferSize,    //出力バッファサイズ
    DWORD      nInBufferSize,     //入力バッファサイズ
```

```
DWORD                          nDefaultTimeOut,         //タイムアウト間隔
LPSECURITY_ATTRIBUTES  lpSecurityAttributes  //SECURITY_ATTRIBUTES
                                                                   構造体へのポインタ
);
```

V B

Declare Function CreateNamedPipe Lib "kernel32" Alias "CreateNamedPipeA" (ByVal lpName As String, ByVal dwOpenMode As Long, ByVal dwPipeMode As Long, ByVal nMaxInstances As Long, ByVal nOutBufferSize As Long, ByVal nInBufferSize As Long, ByVal nDefaultTimeOut As Long, lpSecurityAttributes As SECURITY_ATTRIBUTES) As Long

パラメータ

・*lpName* 名前付きパイプの名称を格納したNULLで終わる文字列へのポインタを指定する。

・*dwOpenMode* パイプハンドルのパイプアクセスモードを指定する。双方向モード、ライトスルーモード、およびセーフティアクセスモードを指定する。
このパラメータは次のパイプアクセスモードフラグのうちの1つ指定する。

値	意味
PIPE_ACCESS_DUPLEX	双方向モードで作成する
PIPE_ACCESS_INBOUND	クライアントからサーバーにだけ通信を行う
PIPE_ACCESS_OUTBOUND	サーバーからクライアントにだけ通信を行う

また、ライトスルーか、オーバーラップかの指定を行う。

値	意味
FILE_FLAG_WRITE_THROUGH	ライトスルーモードである
FILE_FLAG_OVERLAPPED	オーバーラップモードである

セーフティアクセスモードの設定を次の中から指定する。

値	意味
WRITE_DAC	呼び出し側は任意のアクセスを可能にして、アクセスコントロールリストを制御する。
WRITE_OWNER	呼び出し側は名前付きパイプの所有者に対して書き込みが許される。
ACCESS_SYSTEM_SECURITY	呼び出し側はシステムアクセスコントロールリストに対して書き込みが許される。

・*dwPipeMode* 次のタイプモードのいずれかを指定する。
・書き込みモードの場合は、次の値のいずれかを指定する。

値	意味
PIPE_TYPE_BYTE	データはバイトの流れとしてパイプに書かれる。PIPE_READMODE_MESSAGEと同時に指定はできない。

PIPE_TYPE_MESSAGE	データはメッセージの流れとしてパイプに書かれる。PIPE_TYPE_BYTEと同時に指定はできない。

・読み込みモードの場合は、次の値のいずれかを指定する。

値	意味
PIPE_READMODE_BYTE	データはバイトの流れとして読まれる。 PIPE_TYPE_MESSAGE,PIPE_TYPE_BYTEと同時に指定はできない。
PIPE_READMODE_MESSAGE	データはメッセージの流れとして読まれる。 PIPE_TYPE_MESSAGEが指定されている場合のみ有効である。

・ウェイトモードの場合は、次の値のいずれかを指定する。

値	意味
PIPE_WAIT	ブロックモードで動作する。このモードの使用は、クライアントプロセスが動作を実行する状況において無制限に待つことを意味している。
PIPE_NOWAIT	ノンブロックモードで動作する。

・*nMaxInstances*　パイプを利用するアプリケーションインスタンスの最大値を指定する。
・*nOutBufferSize*　出力バッファのサイズをバイトで指定する。
・*nInBufferSize*　入力バッファのサイズをバイトで指定する。
・*nDefaultTimeOut*　タイムアウトの発生時間をミリ秒で指定する。
・*lpSecurityAttributes*　SECURITY_ATTRIBUTES構造体へのポインタを指定する。

戻り値

正常終了	名前付きパイプのサーバー終端を識別するハンドル
異常終了	INVALID_HANDLE_VALUE

拡張エラー情報はGetLastError関数で取得する。

解説

新しい名前付きパイプが作成された場合、パラメーターからのアクセスコントロールリスト（ACL）は、名前付きパイプのための任意のアクセスコントロールを定義する。

参照

→ConnectNamedPipe,CreateFile,→ReadFile,ReadFileEx,
→SECURITY_ATTRIBUTES,TransactNamedPipe,WaitNamedPipe,WriteFile,
WriteFileEx

CreatePipe	kernel32.dll		NT
名前なしパイプを作成			

V C

```
BOOL CreatePipe(
    PHANDLE      hReadPipe,      //入力用パイプを識別するハンドル
    PHANDLE      hWritePipe,     //出力用パイプを識別するハンドル
    LPSECURITY_ATTRIBUTES        lpPipeAttributes,
                                 //SECURITY_ATTRIBUTES構造体へのポインタ
    DWORD        nSize           //パイプに確保されるバッファサイズ
    );
```

V B

```
Declare Function CreatePipe Lib "kernel32" Alias "CreatePipe" (phReadPipe As Long, phWritePipe As Long, lpPipeAttributes As SECURITY_ATTRIBUTES, ByVal nSize As Long) As Long
```

パラメータ

- *hReadPipe*　入力用パイプを識別するハンドルを格納する変数へのポインタを指定する。
- *hWritePipe*　出力用パイプを識別するハンドルを格納する変数へのポインタを指定する。
- *lpPipeAttributes*　SECURITY_ATTRIBUTES構造体へのポインタを指定する。
- *nSize*　パイプのために確保されるバッファサイズを指定する。パラメータに0が指定された場合は、デフォルトのバッファサイズが利用される。

戻り値

正常終了	TRUE
異常終了	FALSE

拡張エラー情報はGetLastError関数で取得する。

解　説

CreatePipe関数は、指定されたパイプサイズを割り当ててパイプを作成する。

参　照

→ReadFile,→SECURITY_ATTRIBUTES,WriteFile

DisconnectNamedPipe	kernel32.dll	95	NT
名前付きパイプの接続を切断			

V　C

```
BOOL DisconnectNamedPipe(
    HANDLE hNamedPipe       //名前付きパイプを識別するハンドル
    );
```

V　B

Declare Function DisconnectNamedPipe Lib "kernel32" Alias "DisconnectNamedPipe" (ByVal hNamedPipe As Long) As Long

パラメータ

・*hNamedPipe*　　名前付きパイプを識別するためのハンドルを指定する。パラメータはCreateNamedPipe関数によって戻される値を利用する。

戻り値

正常終了	TRUE
異常終了	FALSE

拡張エラー情報はGetLastError関数で取得する。

解　説

サーバープロセスは、ハンドルがConnectNamedPipe関数を使って別のクライアントと接続する前にDisconnectNamedPipe関数を呼んで、パイプハンドルをその前のクライアントから切り離さなければならない。

参　照

→CloseHandle,→ConnectNamedPipe,→CreateNamedPipe,→FlushFileBufferst

Process and Thread

CreateProcess	kernel32.dll		NT
プロセスの起動			

V C

```
BOOL CreateProcess(
    LPCTSTR                  lpApplicationName,   //実行可能モジュール
                                                    へのポインタ
    LPTSTR                   lpCommandLine,       //コマンドライン文字列
    LPSECURITY_ATTRIBUTES    lpProcessAttributes,
                      //SECURITY_ATTRIBUTES構造体へのポインタ
    LPSECURITY_ATTRIBUTES    lpThreadAttributes,
                      //SECURITY_ATTRIBUTES構造体へのポインタ
    BOOL                     bInheritHandles,     //ハンドルフラグ
    DWORD                    dwCreationFlags,     //作成フラグ
    LPVOID                   lpEnvironment,       //環境変数へのポインタ
    LPCTSTR                  lpCurrentDirectory,  //カレントディレクト
                                                    リ名へのポインタ
    LPSTARTUPINFO            lpStartupInfo,       //STARTUPINFO構造
                                                    体へのポインタ
    LPPROCESS_INFORMATION    lpProcessInformation
                      //PROCESS_INFORMATION構造体へのポインタ
);
```

V B

```
Declare Function CreateProcess Lib "kernel32" Alias "CreateProcessA" (ByVal lpApplicationName As String, ByVal lpCommandLine As String, lpProcessAttributes As SECURITY_ATTRIBUTES,
```

パラメータ

- *lpApplicationName* 実行可能モジュールを識別するNULLで終わる文字列へのポインタを指定する。
- *lpCommandLine* コマンドラインを格納するNULLで終わる文字列へのポインタを指定する。ファイル名にフルパス名が指定されなかった場合、次の順序でファイルを探す。
  1. アプリケーションのロードディレクトリ
  2. カレントディレクトリ
  3. Windows 95の場合は、GetSystemDirectory関数で取得できるシステムディレクトリ。WindowsNTの場合は、GetSystemDirectory関数で取得できる32ビット用システムディレクトリ。

4. WindowsNTのみ、16ビット用のシステムディレクトリ。
5. GetWindowsDirectory関数で取得できるWindowsディレクトリ。
6. パス環境変数に記述されたディレクトリ。

- *lpProcessAttributes*　作成されるプロセスの情報を格納したSECURITY_ATTRIBUTES構造体へのポインタ。
- *lpThreadAttributes*　作成されるスレッドの情報を格納したSECURITY_ATTRIBUTES構造体へのポインタ。
- *bInheritHandles*　次の値のいずれかを指定する。

値	意味
TRUE	呼び出し元のハンドルを引き継ぐ
FALSE	新しいハンドルが割り当てられる

- *dwCreationFlags*　プロセスのプライオリティクラスと作成を制御する付加的なフラグを指定する。次のフラグのいずれかを指定する。

値	意味
CREATE_DEFAULT_ERROR_MODE	新しいプロセスは呼び出しプロセスのエラーモードを引き継がない。代わりにCreateProcess関数は、新しいプロセスに現在のデフォルトエラーモードを与える。
CREATE_NEW_CONSOLE	新しいプロセスは、親のコンソールを引き継ぐ代わりに新しいコンソールを持つ。 DETACHED_PROCESSと同時に指定できない。
CREATE_NEW_PROCESS_GROUP	新しいプロセスは新しいプロセスグループのベースプロセスである。プロセスグループは、CTRL+CまたはCTRL+BREAKシグナルを1グループのコンソールプロセスに送ることを可能にするために、GenerateConsoleCtrlEvent関数で利用される。
CREATE_SEPARATE_WOW_VDM	新しいプロセスは事実上DOSマシン（VDM）として動作する。
CREATE_SHARED_WOW_VDM	Windows NT：WIN.INIの[WINDOWS]セクションのDefaultSeparateVDMスイッチがTRUEの場合、このフラグはCreateProcess関数のスイッチを無効にして、プロセスが分割された事実上の新しいDOSマシンを起動する。
CREATE_SUSPENDED	新しいプロセスのスレッドは、保留状態で作成されて、ResumeThread関数が呼ばれるまで動かない。

CREATE_UNICODE_ENVIRONMENT	*lpEnvironment*により指し示された環境変数ブロックがUnicode文字を使う。
DEBUG_PROCESS	呼び出しプロセスはデバッガとみなされて、新しいプロセスは、バグを取り除いているプロセスである。呼び出しスレッド（CreateProcessを呼んだスレッド）が、WaitForDebugEvent関数を呼ぶことができない限り、システムはプロセスに存在するすべてのデバッグ事象をデバッガに通知する。
DEBUG_ONLY_THIS_PROCESS	このフラグが設定されず、呼び出しプロセスをデバッグしている場合、新しいプロセスは、呼び出しプロセスによって別のデバッガプロセスになる。
DETACHED_PROCESS	コンソールプロセスのため、新しいプロセスは親プロセスのコンソールに入出力できない。CREATE_NEW_CONSOLEと同時には指定できない。

*dwCreationFlags*パラメータは新しいプロセスを制御する。つぎのプライオリティクラスフラグの値をどれも指定しない場合、作成プロセスのプライオリティクラスがIDLE_PRIORITY_CLASSでない限り、プライオリティクラスはNORMAL_PRIORITY_CLASSとなる。この場合、子プロセスのデフォルトプライオリティクラスはIDLE_PRIORITY_CLASSである。

値	意味
HIGH_PRIORITY_CLASS	このプライオリティクラスプロセスのスレッドは、通常のプライオリティまたはアイドルプライオリティクラスプロセスのスレッドに優先して実行される。
IDLE_PRIORITY_CLASS	スレッドはシステムがアイドルで、より高いプライオリティクラスで動作するプロセスのスレッドに実行順位を先取りされる。
NORMAL_PRIORITY_CLASS	特にスケジューリングする必要のない、通常のプロセスを示す。
REALTIME_PRIORITY_CLASS	最も高いプライオリティを持っているプロセスを示す。リアルタイムプライオリティクラスプロセスのスレッドは他のあらゆるプロセスのスレッドに優先する。

- *lpEnvironment*　パラメータがNULLの場合、新しいプロセスは呼び出しプロセスの環境変数を使う。
- *lpCurrentDirectory*　カレントディレクトリを識別するフルパス名が格納されたNULLで終わる文字列へのポインタを指定する。
- *lpStartupInfo*　新規プロセスのメインウィンドウの情報を格納したSTARTUPINFO構造体へのポインタを指定する。
- *lpProcessInformation*　新規プロセスの属性を格納するPROCESS_INFORMATION構造体へのポインタを指定する。

戻り値

正常終了	TRUE
異常終了	FALSE

拡張エラー情報はGetLastError関数で取得する。

解説

CreateProcess関数は、新しいプログラムを実行するために用いられる。WinExecとLoadModule関数も利用可能だが、これらの呼び出しはCreateProcessによって実行される。プロセスを発生させるだけでなく、CreateProcess関数はスレッドオブジェクトも作成する。スレッドは、サイズが指定されるプログラムの実行可能なファイルのイメージヘッダーにおいて記述されている初期のスタックで作成される。

参照

AllocConsole,CloseHandle,CreateRemoteThread,→CreateThread,ExitProcess,ExitThread,GenerateConsoleCtrlEvent,GetCommandLine,GetEnvironmentStrings,GetExitCodeProcess,→GetFullPathName,GetStartupInfo,→GetSystemDirectory,→GetWindowsDirectory,→LoadModule,OpenProcess,→PROCESS_INFORMATION,ResumeThread,→SECURITY_ATTRIBUTES,SetConsoleCtrlHandler,→SetErrorMode,STARTUPINFO,TerminateProcess,WaitForInputIdle,WaitForDebugEvent,→WinExect

CreateThread	————————	95	NT
スレッドオブジェクトの作成			

V　C

```
HANDLE CreateThread(
    LPSECURITY_ATTRIBUTES    lpThreadAttributes,
                            //SECURITY_ATTRIBUTES構造体へのポインタ
    DWORD                    dwStackSize,       //初期スタックサイズ
    LPTHREAD_START_ROUTINE   lpStartAddress,    //スレッド関数へのポインタ
    LPVOID                   lpParameter,       //パラメータ
    DWORD                    dwCreationFlags,   //作成フラグ
    LPDWORD                  lpThreadId         //スレッド識別子を格納す
                                                  る変数へのポインタ
    );
```

パラメータ

- *lpThreadAttributes*　作成するスレッド情報を格納したSECURITY_ATTRIBUTES構造体へのポインタ。
- *dwStackSize*　初期のスタックサイズをバイトで指定する。
- *lpStartAddress*　スレッドにより実行される関数を識別するアドレスへのポインタを指定する。
- *lpParameter*　スレッドに引き渡されるパラメータの値を指定する。
- *dwCreationFlags*　新しいスレッドの作成時に適用されるスレッドの属性を指定する。

値	意味
CREATE_DEFAULT_ERROR_MODE	新しいプロセスは呼び出しプロセスのエラーモードを引き継がない。代わりにCreateProcess関数は、新しいプロセスに現在のデフォルトエラーモードを与える。
CREATE_NEW_CONSOLE	新しいプロセスは、親のコンソールを引き継ぐ代わりに新しいコンソールを持つ。 DETACHED_PROCESSと同時に指定できない。
CREATE_NEW_PROCESS_GROUP	新しいプロセスは新しいプロセスグループのベースプロセスである。プロセスグループは、CTRL+CまたはCTRL+BREAKシグナルを1グループのコンソールプロセスに送ることを可能にするために、GenerateConsoleCtrlEvent関数で利用される。
CREATE_SEPARATE_WOW_VDM	新しいプロセスは事実上DOSマシン（VDM）として動作する。
CREATE_SHARED_WOW_VDM	Windows NT：WIN.INIの[WINDOWS]セクションのDefaultSeparateVDMスイッチがTRUEの場合、このフラグはCreateProcess関数のスイッチを無効にして、プロセスの分割された事実上の新しいDOSマシンを起動する。
CREATE_SUSPENDED	新しいプロセスのスレッドは、保留状態で作成されて、ResumeThread関数が呼ばれるまで動かない。
CREATE_UNICODE_ENVIRONMENT	*lpEnvironment*により指し示された環境変数ブロックがUnicode文字を使う。

・*lpThreadId*　作成されたスレッドの識別子を格納する変数へのポインタを指定する。

戻り値

正常終了	スレッドオブジェクトのハンドル
異常終了	NULL

拡張エラー情報はGetLastError関数で取得する。

参　照

→CloseHandle,→CreateProcess,CreateRemoteThread,ExitProcess,ExitThread,
GetExitCodeThread,GetThreadPriority,ResumeThread,SetThreadPriority,
→SECURITY_ATTRIBUTES

GetCurrentProcess	kernel32.dll	95	NT
現在のプロセスを取得			

V C　HANDLE GetCurrentProcess()

V B　Declare Function GetCurrentProcess Lib "kernel32" Alias "GetCurrentProcess" () As Long

戻り値　現在のプロセスを識別するpseudohandle。
pseudohandleは、現在のプロセスハンドルと解釈される特別な定数である。プロセスハンドルが必要な場合、呼び出しプロセスはいつでも自分自身のプロセスを指定するために、このハンドルを用いることができる。pseudohandleは子プロセスに引き継がれない。

参　照　→CloseHandle,DuplicateHandle,GetCurrentProcessId,→GetCurrentThread,OpenProcess

GetCurrentThread	kernel32.dll	95	NT
現在のスレッドオブジェクトを取得			

V C　HANDLE GetCurrentThread()

V B　Declare Function GetCurrentThread Lib "kernel32" Alias "GetCurrentThread" () As Long

戻り値　現在のスレッドを識別するpseudohandle。
pseudohandleは、現在のスレッドハンドルと解釈される特別な定数である。スレッドハンドルが必要な場合、呼び出しプロセスはいつでもそれ自身のスレッドを指定するために、このハンドルを用いることができる。pseudohandleは子プロセスに引き継がれない。

参　照　CloseHandle,DuplicateHandle,GetCurrentProcess,GetCurrentThreadId

WinExec	kernel32.dll	95	NT
ほかのアプリケーションを実行			

V C

```
UINT WinExec (
    LPCSTR  lpCmdLine,      //コマンドライン文字列へのポインタ
    UINT    uCmdShow        //アプリケーション起動時のスタイル
    );
```

V B

```
Declare Function WinExec Lib "kernel32" Alias "WinExec" (ByVal lpCmdLine As String, ByVal nCmdShow As Long) As Long
```

パラメータ

・*lpCmdLine*　アプリケーションを実行するためのコマンドライン（ファイル名とオプションのパラメータ）を格納したNULLで終わる文字列へのポインタ。ファイル名、パラメータともに全角文字を指定することが可能。ファイル名にパス名が含まれない場合、関数は次の順序でディレクトリを検索する。

1. 呼び出し元アプリケーションがロードされたディレクトリ
2. カレントディレクトリ
3. Windowsシステムディレクトリ。GetSystemDirectory関数で取得できるディレクトリ
4. Windowsディレクトリ。GetWindowsDirectory関数で取得できるディレクトリ
5. PATH環境変数内にリストされているディレクトリ

・*uCmdShow*　アプリケーション起動時のスタイルを指定する。非Windowsアプリケーションの場合は、アプリケーション用のプログラム情報ファイル（PIF）によりウィンドウの状態が決定される。*uCmdShow*パラメータには次の値のいずれかを指定する。

値	意味
SW_HIDE	ウィンドウを非表示にし、ほかのウィンドウをアクティブ化する
SW_MAXIMIZE	指定されたウィンドウを最大化
SW_MINIMIZE	指定されたウィンドウを最小化し、タスクリストのトップに位置させる
SW_RESTORE	ウィンドウをアクティブ化し表示。ウィンドウが最小化または最大化されている場合には元のサイズと位置に復元(SW_SHOWNORMALと等価)
SW_SHOW	ウィンドウをアクティブ化し現在のサイズと位置で表示
SW_SHOWDEFAULT	STARTUPINFO構造体の情報を利用する
SW_SHOWMAXIMIZED	ウィンドウをアクティブ化し最大化表示する
SW_SHOWMINIMIZED	ウィンドウをアクティブ化し最小化表示する
SW_SHOWMINNOACTIVE	ウィンドウを最小化。現在アクティブなウィンドウはアクティブな状態を維持する
SW_SHOWNA	ウィンドウを現在の状態で表示。現在アクティブなウィンドウはアクティブな状態を維持する
SW_SHOWNOACTIVATE	以前に表示されたサイズと位置でウィンドウを表示。現在アクティブなウィンドウはアクティブな状態を維持する
SW_SHOWNORMAL	ウィンドウをアクティブ化し表示。ウィンドウが最小化または最大化されている場合には元のサイズと位置に復元

戻り値

正常終了	31以上
異常終了	次の値のいずれかを返す

値	意味
0	メモリもしくはリソースが不足している
ERROR_BAD_FORMAT	指定されたファイルが実行可能形式のファイルではない
ERROR_FILE_NOT_FOUND	指定されたファイルが見つからない
ERROR_PATH_NOT_FOUND	指定されたパス名が正しくない

参　照

CreateProcess,GetMessage,GetSystemDirectory,→GetWindowsDirectory,LoadModule,ShowWindow

Registory

RegCreateKey	advapi32.dll	95	NT
キーを作成またはオープン			

V C

```
LONG RegCreateKey(
    HKEY      hKey,       //オープンするキーを識別するハンドル
    LPCTSTR lpSubKey, //オープンもしくは作成するサブキーを指定するNULL
                        で終わる文字列へのポインタ
    PHKEY    phkResult //オープンもしくは作成されたキーのハンドル
    );
```

V B

Declare Function RegCreateKey Lib "advapi32.dll" Alias "RegCreateKeyA" (ByVal hKey As Long, ByVal lpSubKey As String, phkResult As Long) As Long

パラメータ

・*hKey* オープンするキーを識別するハンドルを指定する。パラメータに指定できる値は次の通り予約されている。

HKEY_CLASSES_ROOT
HKEY_CURRENT_USER
HKEY_LOCAL_MACHINE
HKEY_USERS

・*lpSubKey* オープンもしくは作成するサブキーを指定するNULLで終わる文字列へのポインタ。

・*phkResult* オープンもしくは作成されたキーのハンドル。

戻り値

正常終了	ERROR_SUCCESS
異常終了	エラー値

解説

アプリケーションは*hKey*パラメータにHKEY_CLASSES_ROOTを指定することで、トップレベルのデータベースに従属するキーを作成することができる。また、RegCreateKey関数を使用して、一度に複数のキーを作成することもできる。*lpSubKey*パラメータに次の形式の文字列を指定するとレベルの深さが4で、先行する3つのサブキーを持つサブキーを作成することができる。

subkey1\subkey2\subkey3\subkey4

参照

RegCloseKey,→RegCreateKeyEx,→RegDeleteKey,RegOpenKey,RegOpenKeyEx,
→RegSetValue

RegDeleteKey	advapi32.dll	95	NT
キーを削除			

V C

```
LONG RegDeleteKey(
    HKEY      hKey,       //オープンするキーを識別するハンドル
    LPCTSTR lpSubKey  //削除するサブキーを指定するNULLで終わる文字列へ
                          のポインタ
    );
```

V B

Declare Function RegDeleteKey Lib "advapi32.dll" Alias "RegDeleteKeyA" (ByVal hKey As Long, ByVal lpSubKey As String) As Long

パラメータ

・*hKey*　オープンするキーを識別するハンドルを指定する。パラメータに指定できる値は次の通り予約されている。

HKEY_CLASSES_ROOT
HKEY_CURRENT_USER
HKEY_LOCAL_MACHINE
HKEY_USERS

・*lpSubKey*　削除するサブキーを指定するNULLで終わる文字列へのポインタを指定する。

戻り値

正常終了	ERROR_SUCCESS
異常終了	エラー値

解　説

WindowsNTではRegDeleteKey関数を利用して、サブキーを持つサブキーを削除することはできない。

参　照

RegCloseKey,→RegCreateKeyEx,RegOpenKeyEx

RegSetValue	advapi32.dll	95	NT
指定されたキーにテキスト文字列を関連付ける			

V C

```
LONG RegSetValue(
    HKEY     hKey,      //値を設定するキーを識別するハンドル
    LPCTSTR  lpSubKey,  //値を設定するサブキーを識別するNULLで終わる文字
                          列へのポインタ
    DWORD    dwType,    //設定する値のタイプ
    LPCTSTR  lpData,    //設定する値を格納したNULLで終わる文字列へのポイ
                          ンタ
    DWORD    cbData     //設定する値のサイズ
);
```

V B

Declare Function RegSetValue Lib "advapi32.dll" Alias "RegSetValueA" (ByVal hKey As Long, ByVal lpSubKey As String, ByVal dwType As Long, ByVal lpData As String, ByVal cbData As Long) As Long

パラメータ

- *hKey*　オープンするキーを識別するハンドルを指定する。パラメータに指定できる値は次の通り予約されている。
  HKEY_CLASSES_ROOT
  HKEY_CURRENT_USER
  HKEY_LOCAL_MACHINE
  HKEY_USERS
- *lpSubKey*　値を設定するサブキーを識別するNULLで終わる文字列へのポインタ。
- *dwType*　設定する値のタイプ。REG_SZタイプを指定しなければならない。他のタイプのキーを設定する場合には、RegSetValueEx関数を利用する。
- *lpData*　設定する値を格納したNULLで終わる文字列へのポインタを指定する。
- *cbData*　設定する値のサイズを指定するが、これに終端のNULL文字は含まれない。

戻り値

正常終了	ERROR_SUCCESS
異常終了	エラー値

解　説

*lpSubKey*パラメータで指定されたキーが存在しないときは、RegSetValue関数がキーを作成する。RegOpenKey関数を使ってオープンされてかつRegCloseKey関数の呼び出しでクローズされるキーに対してRegSetValue関数の呼び出しを実行するとシステムの処理速度が向上する。

参　照

RegCreateKeyEx,RegFlushKey,RegOpenKeyEx,RegQueryValue,RegQueryValueEx,RegSetValueEx

Resource

LoadImage	user32.dll	95	NT
リソースからビットマップイメージをロード			

VC

```
HANDLE LoadImage(
    HINSTANCE  hinst,       //イメージが格納されているアプリケーションの
                              インスタンスを識別するハンドル。
    LPCTSTR    lpszName,    //ビットマップイメージのリソース識別子
    UINT       uType,       //ビットマップイメージタイプ
    int        cxDesired,   //イメージの幅
    int        cyDesired,   //イメージの高さ
    UINT       fuLoad       //ロードフラグ
    );
```

VB

```
Declare Function LoadImage Lib "user32" Alias "LoadImageA" (ByVal hInst As Long, ByVal lpsz As String, ByVal un1 As Long, ByVal n1 As Long, ByVal n2 As Long, ByVal un2 As Long) As Long
```

パラメータ

・*hinst*　ロードするイメージを含むモジュールのインスタンスを識別するハンドルを指定する。OEMイメージをロードするには、このパラメータをを0に設定する。

・*lpszName*　ロードするイメージを識別する文字列へのポインタを指定する。*hinst*パラメータがNULL以外で、*fuLoad*パラメータがLR_LOADFROMFILEを含まない場合、*lpszName*は*hinst*モジュールのイメージリソースの名前を含むNULLで終わる文字列へのポインタである。*hinst*がNULLで、LR_LOADFROMFILEが指定されていない場合、このパラメータの下位ワードは、ロードするOEMイメージの識別子である。OEMイメージ識別子はWINUSER.Hで定義され、次の接頭辞を持つ。

接頭辞	意味
OBM_	OEMビットマップ
OIC_	OEMアイコン
OCR_	OEMカーソル

Windows 95では、*fuLoad*パラメータがLR_LOADFROMFILE値を含んでいる場合、*lpszName*はイメージを含んでいるファイル名である。Windows NTでは、LR_LOADFROMFILEは無視される。

・*uType*　ロードするイメージのタイプを指定する。次の値のいずれかを指定する。

値	意味
IMAGE_BITMAP	ビットマップ
IMAGE_CURSOR	カーソル
IMAGE_ICON	アイコン

・*cxDesired*　　ロードされるイメージの幅を指定する。
・*cyDesired*　　ロードされるイメージの高さを指定する。
・*fuLoad*　　次の値の組み合わせを指定する。

値	意味
LR_DEFAULTCOLOR	デフォルト。なにも処理はしない。
LR_CREATEDIBSECTION	Windows 95。*uType*パラメータがIMAGE_BITMAPを指定している場合、DIBセクションビットマップを戻す。表示装置の色にあわせたカラーマップを作らずにビットマップをロードする場合に有効。
LR_DEFAULTSIZE	Windows 95。*cxDesired*、*cyDesired*の値が0に設定されている場合、カーソルとアイコンのためのシステムメトリクス値に基づいた幅、高さを設定する。
LR_LOADFROMFILE	Windows 95。*lpszName*パラメータで指定されるファイルからイメージをロードする。
LR_LOADMAP3DCOLORS	Windows 95。イメージのためのカラーテーブルおよびグレースケールのシェードを3Dカラーと交換する。対応は「●シェード対3Dカラー」を参照。
LR_LOADTRANSPARENT	Windows 95。イメージの最初のピクセルのカラー値を取り出し、カラーテーブルのデフォルトウィンドウ色と交換する（COLOR_WINDOW）。
LR_MONOCHROME	Windows 95。白黒イメージでロードする。
LR_SHARED	Windows 95。イメージを共有可能な状態でロードする。

●シェード対3Dカラー

色	交換色
Dk Gray,RGB(128,128,128)	COLOR_3DSHADOW
Gray,RGB(192,192,192)	COLOR_3DFACE
Lt Gray,RGB(223,223,223)	COLOR_3DLIGHT

戻り値

正常終了	ロードされたイメージのハンドル
異常終了	NULL

参照

CopyImage,→GetSystemMetrics,→LoadBitmap,→LoadCursor,→LoadIcont

LoadResource	kernel32.dll	95	NT
リソースをロード			

V　C

```
HGLOBAL LoadResource(
    HMODULE    hModule,  //ロードするリソースを持つ実行可能モジュールのハンドル
    HRSRC      hResInfo  //ロードするリソースのハンドル
    );
```

V　B

Declare Function LoadResource Lib "kernel32" Alias "LoadResource" (ByVal hInstance As Long, ByVal hResInfo As Long) As Long

パラメータ

・*hModule*　ロードするリソースを持つ実行可能モジュールのハンドルを指定する。NULLを指定した場合は、現在のプロセスのモジュールが参照の対象となる。

・*hResInfo*　ロードするリソースのハンドルを指定する。このハンドルはFindResourceもしくはFindResourceEx関数により取得する。

戻り値

正常終了	ロードされたリソースが割り当てられたグローバルメモリブロックのハンドル
異常終了	NULL

拡張エラー情報はGetLastError関数で取得。

解　説

アプリケーションはリソースの使用を終えたら、関連するグローバルメモリをFreeResource関数を使用して解放しなければならない。指定されたリソースがすでにロードされている場合には、関数はリソースの参照カウントを増加させるだけである。

参　照

FindResource,FindResourceEx,FreeResource,LoadLibrary,LoadModule,LockResource

String Manipulation

CharNext	user32.dll	95	NT
文字列内の次の文字に移動			

V　C

```
LPCSTR CharNext(
   LPCTSTR  lpsz //現在のキャラクタを格納するNULLで終わる文字列へのポインタ
   );
```

V　B

Declare Function CharNext Lib "user32" Alias "CharNextA" (ByVal lpsz As String) As String

パラメータ　・*lpsz*　現在のキャレット位置にある文字を示すNULLで終わる文字列へのポインタを指定する。

戻り値		
	正常終了	次の文字へのポインタもしくはNULL文字

参　照　Char Next ExA,Char Prev

CharUpper	user32.dll	95	NT
文字を大文字に変換			

V　C

```
LPTSTR CharUpper(
    LPTSTR    lpsz        //文字もしくは文字列へのポインタ
    );
```

V　B

```
Declare Function CharUpper Lib "user32" Alias "CharUpperA" (ByVal lpsz As String)
As String
```

パラメータ　・*lpsz*　NULLで終わる文字列へのポインタもしくは1文字を指定する。上位ワードが0の場合、下位ワードには単一の1文字が指定される。

戻り値　変換後の文字列へのポインタ。

参　照　→CharLower,CharLowerBuff,CharUpperBuff

CompareString	kernel32.dll	95	NT
2つの文字列を比較			

V　C

```
int CompareString(
    LCID       Locale,        //環境識別子
    DWORD      dwCmpFlags,    //比較スタイル
    LPCTSTR    lpString1,     //第1文字列へのポインタ
    int        cchCount1,     //第1文字列のサイズ
    LPCTSTR    lpString2,     //第2文字列へのポインタ
    int        cchCount2      //第2文字列のサイズ
    );
```

V　B

```
Declare Function CompareString Lib "kernel32" Alias "CompareStringA" (ByVal
Locale As Long, ByVal dwCmpFlags As Long, ByVal lpString1 As String, ByVal
cchCount1 As Long, ByVal lpString2 As String, ByVal cchCount2 As Long) As Long
```

パラメータ

・*Locale*　　次の識別子のいずれかを指定する。

値	意味
LOCALE_SYSTEM_DEFAULT	システムデフォルト
LOCALE_USER_DEFAULT	ユーザーデフォルト

・*dwCmpFlags*　比較の方法を指定するフラグを設定する。デフォルトではフラグは設定されていない。

値	意味
NORM_IGNORECASE	大文字と小文字の比較はしない
NORM_IGNOREKANATYPE	ひらがなとカタカナを区別しない
NORM_IGNORENONSPACE	スペースを無視
NORM_IGNORESYMBOLS	記号を無視
NORM_IGNOREWIDTH	1バイト文字と2バイト文字を区別しない
SORT_STRINGSORT	句読点を記号として扱う

・*lpString1*　　第1文字列へのポインタを指定する。
・*cchCount1*　　第1文字列のサイズを指定する。
・*lpString2*　　第2文字列へのポインタを指定する。
・*cchCount2*　　第2文字列のサイズを指定する。

戻り値

次のいずれかの戻り値を返す。

1	*lpString1*パラメータは、*lpString2*パラメータより語数が少ない。
2	*lpString1*パラメータは、*lpString2*パラメータと同じ語数である。
3	*lpString1*パラメータは、*lpString2*パラメータより語数が多い。
0	異常終了

拡張エラー情報はGetLastError関数で取得する。

参　照

FoldString,GetSystemDefaultLCID,GetUserDefaultLCID,LCMapString,lstrcmp,lstrcmpi,
→MAKELCIDt

FormatMessage	kernel32.dll	95	NT
メッセージ形式を設定			

V　C

```
DWORD FormatMessage(
    DWORD       dwFlags,        //処理選択フラグ
    LPCVOID     lpSource,       //メッセージ文字列へのポインタ
    DWORD       dwMessageId,    //メッセージIDを識別
    DWORD       dwLanguageId,   //メッセージ言語を識別
    LPTSTR      lpBuffer,       //メッセージの格納されるバッファへのポインタ
    DWORD       nSize,          //メッセージバッファのサイズ
    va_list     *Arguments      //メッセージ配列のアドレス
    );
```

V　B

```
Declare Function FormatMessage Lib "kernel32" Alias "FormatMessageA" (ByVal dwFlags As Long, lpSource As Any, ByVal dwMessageId As Long, ByVal dwLanguageId As Long, ByVal lpBuffer As String, ByVal nSize As Long, Arguments As Long) As Long
```

パラメータ

・*dwFlags*　整形の方法を指定する*lpSource*パラメータをどのように編集するかの1セットのビットフラグを含む。*dwFlags*の下位バイトは、機能ハンドルラインの出力バッファを指定する。下位バイトは、フォーマットされたアウトプットラインの最大幅を指定している。次の値のいずれかを指定する。

値	意味
FORMAT_MESSAGE_ALLOCATE_BUFFER	フォーマットされたメッセージを保存するのに十分なバッファを割り当て、*lpBuffer*で指定されたアドレスへのポインタを保持する。メッセージが不要になった場合は、バッファを解放するために、LocalFree関数を用いる。
FORMAT_MESSAGE_IGNORE_INSERTS	不変の出力バッファであることを指定する。
FORMAT_MESSAGE_FROM_STRING	*lpSource*がNULLで終わる文字列へのポインタであることを示す。
FORMAT_MESSAGE_FROM_HMODULE	*lpSource*がメッセージテーブルリソースを含むモジュールハンドルであることを示す。
FORMAT_MESSAGE_ARGUMENT_ARRAY	32ビット値の配列へのポインタであることを示す。

*dwFlags*の下位バイトは、整形されたアウトプットラインの最大幅を指定する。次の値のいずれかを指定する。

値	意味
0	アウトプット行幅に制限がない。
0以外	アウトプットラインの文字の最大数を指定している。
FORMAT_MESSAGE_MAX_WIDTH_MASK	メッセージ定義テキストでのライン破棄を無視する。

・*lpSource*　メッセージ定義の位置を指定する。このパラメータのタイプは*dwFlags*パラメータでのセッティングに依存する。*dwFlags*の設定が、FORMAT_MESSAGE_FROM_HMODULE,FORMAT_MESSAGE_FROM_STRING以外の場合、*lpSource*は無視される。

・*dwMessageId*　要求されたメッセージのメッセージ識別子を指定する。*dwFlags*がFORMAT_MESSAGE_FROM_STRINGを含む場合、このパラメータは無視される。

・*dwLanguageId*　要求されたメッセージの言語識別子を指定する。*dwFlags*がFORMAT_MESSAGE_FROM_STRINGを含む場合、このパラメータは無視される。

・*lpBuffer*　メッセージのためのNULLで終わる文字列バッファへのポインタ。*dwFlags*がFORMAT_MESSAGE_ALLOCATE_BUFFERを含む場合、関数は、LocalAlloc関数を使ってバッファを割り当て、指定されたアドレスバッファを*lpBuffer*に設定する。

・*nSize*　FORMAT_MESSAGE_ALLOCATE_BUFFERが設定されていた場合、このパラメータは出力バッファに保存されるバイト（ANSIバージョン）または文字（Unicodeバージョン）の最大数を指定する。

・*Arguments*　メッセージに渡されるパラメータの値を指定する。%1は第1パラメータを%2は第2パラメータを指定する。

戻り値

正常終了	出力バッファに返されるバイト数(ASCII)もしくは文字(Unicode)
異常終了	0

拡張エラー情報はGetLastError関数で取得する。

解　説

メッセージテキスト内には、メッセージを動的に整形するため、いくつかのエスケープシーケンスが利用される。

参　照

→LoadString,LocalFree

GetTimeFormat | kernel32.dll | 95 | NT

時間文字列形式を取得

VC

```
int GetTimeFormat(
    LCID                Locale,     //整形される環境
    DWORD               dwFlags,    //整形の方法
    CONST SYSTEMTIME    *lpTime,    //SYSTEMTIME構造体へのポインタ
    LPCTSTR             lpFormat,   //時間形式文字列へのポインタ
    LPTSTR              lpTimeStr,  //整形された文字列を格納するバッファへ
                                     のポインタ
    int                 cchTime     //整形された文字列を格納するバッファサ
                                     イズ
    );
```

VB

```
Declare Function GetTimeFormat Lib "kernel32" Alias "GetTimeFormatA" (ByVal Locale As Long, ByVal dwFlags As Long, lpTime As SYSTEMTIME, ByVal lpFormat As String, ByVal lpTimeStr As String, ByVal cchTime As Long) As Long
```

パラメータ

・*Locale*　時間文字列を整形する環境を次から選択して指定する。

値	意味
LOCALE_SYSTEM_DEFAULT	システムデフォルト
LOCALE_USER_DEFAULT	ユーザーデフォルト

・*dwFlags* 書式化の方法を指定するフラグを次のいずれかから設定する。

値	意味
LOCALE_NOUSEROVERRIDE	関数は、指定されたシステムデフォルト時間形式を使って文字列を整形する。
TIME_NOMINUTESORSECONDS	分または秒を利用しない。
TIME_NOSECONDS	秒を利用しない。
TIME_NOTIMEMARKER	タイムマーカーを利用しない。
TIME_FORCE24HOURFORMAT	24時間形式。

・*lpTime*　整形される時間情報を含むSYSTEMTIME構造体へのポインタ。NULLの場合は、現在のローカルシステム時間を利用する。

・*lpFormat*　　*lpFormat*がNULLの場合、関数は指定された環境の時間形式を使う。代替文字はそれぞれ次の通りの意味を持つ。

文字	意味
h	12時間形式、0詰めされない「時」
hh	12時間形式、0詰めされた「時」
H	24時間形式、0詰めされない「時」
HH	24時間形式、0詰めされた「時」
m	0詰めされない「分」
mm	0詰めされた「分」
s	0詰めされない「秒」
ss	0詰めされた「秒」
t	AもしくはPで表される、タイムマーカー
tt	AMもしくはPMで表される、タイムマーカー

・*lpTimeStr*　　整形された時間を受け取るための文字列へのポインタを指定する。NULLの場合は、現在のローカルシステム時間を利用する。

・*cchTime*　　*lpTimeStr*のバッファのサイズをバイト(ANSI)もしくは文字(Unicode)で指定する。*cchTime*が0の場合、整形された時間を受け取るために必要なバイト数 もしくは文字数を戻し、*lpTimeStr*により指し示されたバッファは使われない。

戻り値

正常終了	整形された時間文字列のバイト数
異常終了	0

拡張エラー情報はGetLastError関数で取得する

参照

GetDateFormat,→SYSTEMTIME

LoadString	user32.dll	95	NT
文字列リソースをロード			

V C

```
int LoadString(
    HINSTANCE   hInstance,    //文字列リソースを格納しているモジュールインスタンスのハンドル
    UINT        uID,          //ロードする文字列を指定する整数識別子
    LPTSTR      lpBuffer,     //文字列を受け取るバッファへのポインタ
    int         nBufferMax    //バッファのサイズを指定
    );
```

V B

```
Declare Function LoadString Lib "user32" Alias "LoadStringA" (ByVal hInstance As Long, ByVal wID As Long, ByVal lpBuffer As String, ByVal nBufferMax As Long) As Long
```

パラメータ

- *hInstance*　文字列リソースを格納しているモジュールインスタンスを識別するためのハンドル。
- *uID*　ロードする文字列を識別する整数識別子を指定する。
- *lpBuffer*　文字列を受け取るNULLで終わる文字列へのポインタ。
- *nBufferMax*　文字列とで終わるのNULLを格納するのに十分なバッファサイズを指定する。バッファサイズに入りきらない文字列が返された場合には、文字列は切りつめられる。

戻り値

正常終了	文字列のサイズがバイトで返される
異常終了	0

拡張エラー情報はGetLastError関数で取得。

参　照

FormatMessage,LoadAccelerators,→LoadBitmap,LoadCursor,LoadIcon,LoadMenu,LoadMenuIndirect

MultiByteToWideChar	kernel32.dll	95	NT
マルチバイト文字をワイド文字へマップ			

V　C

```
int MultiByteToWideChar(
    UINT      CodePage,          //コードページ
    DWORD     dwFlags,           //キャラクタタイプ
    LPCSTR    lpMultiByteStr,    //マップする文字列へのポインタ
    int       cchMultiByte,      //マップする文字列のサイズ
    LPWSTR    lpWideCharStr,     //ワイド文字を格納する文字列へのポインタ
    int       cchWideChar        //ワイド文字を格納する文字列のバッファサイズ
    );
```

V　B

```
Declare Function MultiByteToWideChar Lib "kernel32" Alias "MultiByteToWideChar"
(ByVal CodePage As Long, ByVal dwFlags As Long, ByVal lpMultiByteStr As String,
ByVal cchMultjByte As Long, ByVal lpWideCharStr As String, ByVal cchWideChar
As Long) As Long
```

パラメータ

・*CodePage*　変換するために用いられるコードページを指定する。次の値は、システムデフォルトコードページのうちの1つを指定するために設定する。

値	意味
CP_ACP	ANSIコードページ
CP_MACCP	Macintoshコードページ
CP_OEMCP	OEMコードページ

・*dwFlags*　マッピングの方法を次のいずれかの値で指定する。

値	意味
MB_PRECOMPOSED	デフォルト。すべての文字とスペース文字はシングル文字としてマップされる。MB_COMPOSITEと同時には指定できない。
MB_COMPOSITE	常にコンポジットキャラクタを利用する。MB_PRECOMPOSEDと同時には指定できない。
MB_ERR_INVALID_CHARS	関数が無効な入力文字見つけた場合、失敗する。GetLastError関数を利用することで、ERROR_NO_UNICODE_TRANSLATIONが返される。
MB_USEGLYPHCHARS	コントロールキャラクタの代わりにglyph文字を使う。

・*lpMultiByteStr*　変換される文字列へのポインタを指定する。

・cchMultiByte	*lpMultiByteStr*パラメータで示された文字列のサイズを指定する。この値が-1の場合、文字列は無効である。NULLで終わる文字の長さは自動的に計算される。
・*lpWideCharStr*	変換後の文字が格納される文字列へのポインタを指定する。
・*cchWideChar*	*lpWideCharStr*パラメータで示された文字列のサイズを指定する。この値が0の場合、関数は*lpWideCharStr*バッファを利用しない。

戻り値

正常終了（cchWideChar = 0以外）	*lpWideCharStr*パラメータに返された文字列のサイズ
正常終了（cchWideChar = 0）	変換後の文字列を受け取るのに必要なバッファサイズ
異常終了	0

拡張エラー情報はGetLastError関数で取得する。

解説

*lpMultiByteStr*と*lpWideCharStr*文字列へのポインタは同じであってはならない。これらが同じ場合、関数は失敗しGetLastErrorはERROR_INVALID_PARAMETERを返す。MB_ERR_INVALID_CHARSが設定されて、マップする文字に無効な文字が見つかった場合も関数は失敗する。

参照

WideCharToMultiByte

Structured Exception Handing

AbnormalTermination

AbnormalTermination	―――――	95	NT

try-finallyの状態を返す

V C　BOOL AbnormalTermination();

戻り値

try-finally ステートメントが異常終了した場合	TRUE
それ以外の場合	FALSE

解説　try ブロック内のすべてのステートメントが順次実行され、最後のステートメントまで到達した場合のみがtry ブロックの正常終了である。例外やreturn、goto、continue、break などのステートメントが実行された場合などはtry ブロックの異常終了と判断する。

参照　MapWindowPoints,POINT,ScreenToClient

GetExceptionCode

GetExceptionCode	―――――	95	NT

例外の種類を返す

V C　DWORD GetExceptionCode();

戻り値　例外のタイプを識別する。次の例外コードのいずれかが返される。

値	意味
EXCEPTION_ACCESS_VIOLATION	スレッドは、不当な仮想アドレスを読み込むもしくは書き込もうとした。
EXCEPTION_BREAKPOINT	ブレークポイントに遭遇した。
EXCEPTION_DATATYPE_MISALIGNMENT	スレッドは、不連続なアドレス空間へのアクセスを試みた。
EXCEPTION_ARRAY_BOUNDS_EXCEEDED	スレッドが、アクセスを禁止されている配列要素にアクセスすることを試みた。
EXCEPTION_FLT_DENORMAL_OPERAND	浮動小数点演算のオペランドの1つが不正。
EXCEPTION_FLT_DIVIDE_BY_ZERO	0の浮動小数点除値で除算を試みた。
EXCEPTION_FLT_INEXACT_RESULT	浮動小数点演算の結果が不正。
EXCEPTION_FLT_INVALID_OPERATION	このリストに含められないあらゆる浮動小数点演算の例外を表す。
EXCEPTION_FLT_OVERFLOW	浮動小数点演算の結果がオーバーフローした。

EXCEPTION_FLT_STACK_CHECK	スタックのオーバーフローもしくは浮動小数点演算の結果がアンダーフローする。
EXCEPTION_FLT_UNDERFLOW	浮動小数点演算の結果が許される値よりも少ない。
EXCEPTION_INT_DIVIDE_BY_ZERO	スレッドは整数の0除算を試みた。
EXCEPTION_INT_OVERFLOW	整数演算の結果が不正。
EXCEPTION_PRIV_INSTRUCTION	スレッドが、現在のマシンモードでは許されない操作を実行しようと試みた。
EXCEPTION_NONCONTINUABLE_EXCEPTION	スレッドは継続不能な例外を起こした後なお、処理を継続しようとした。

解　説

GetExceptionCode関数は、try exceptステートメントのフィルタ表現または例外取り扱いブロック内で呼ばれる。以下にtry exceptステートメントの構造を示す。

```
try{
        /* try block */
}
except (filter-expression) {
        /* exception handler block */
}
```

参　照

→GetExceptionInformation,→RaiseException

GetExceptionInformation	――――――――	95	NT

GetExceptionInformation
例外の詳細情報を返す

V　C

```
LPEXCEPTION_POINTERS GetExceptionInformation();
```

戻り値

例外エラーに関する詳細情報が含まれるEXCEPTION_RECORD構造体と、ハードウェア状態の情報が含まれるCONTEXT構造体へのポインタを格納したEXCEPTION_POINTERS構造体へのポインタが返される。

参　照

→CONTEXT,→EXCEPTION_POINTERS,→EXCEPTION_RECORD,
GetExceptionCode

RaiseException	kernel32.dll	95	NT

RaiseException
例外を発生させる

V　C

```
VOID RaiseException(
    DWORD          dwExceptionCode,       //例外コード
    DWORD          dwExceptionFlags,      //続行可能例外フラグ
    DWORD          nNumberOfArguments,    //配列の要素数
    CONST DWORD    *lpArguments           //文字列配列へのポインタ
    );
```

V　B

```
Declare Sub RaiseException Lib "kernel32" Alias "RaiseException" (ByVal dwExceptionCode As Long, ByVal dwExceptionFlags As Long, ByVal nNumberOfArguments As Long, lpArguments As Long)
```

パラメータ

- *dwExceptionCode*　アプリケーション定義された例外コードを指定する。
- *dwExceptionFlags*　例外フラグを指定する。次の値のいずれかを指定する。

値	意味
続行可能な例外	0
続行不能な例外	EXCEPTION_NONCONTINUABLE

- *nNumberOfArguments*　*lpArguments*配列において要素の数を指定する。EXCEPTION_MAXIMUM_PARAMETERSを越えてはならない。*lpArguments*がNULLの場合パラメータは無視される
- *lpArguments*　32ビット値を持つ配列へのポインタ。

解　説

RaiseException関数は、プロセスが独自に定義したアプリケーション定義の例外を、構造化例外として扱うことを可能にする。

参　照

ExitProcess,→GetExceptionCode,→GetExceptionInformation,WaitForDebugEvent

SetUnhandledExceptionFilter	kernel32.dll	95	NT
例外フィルタ関数を設定			

V C

```
LPTOP_LEVEL_EXCEPTION_FILTER SetUnhandledExceptionFilter(
    LPTOP_LEVEL_EXCEPTION_FILTER    lpTopLevelExceptionFilter
                                        //例外フィルタ関数へのポインタ
    );
```

V B

Declare Function SetUnhandledExceptionFilter Lib "kernel32" Alias "SetUnhandled ExceptionFilter" (ByVal lpTopLevelExceptionFilter As Long) As Long

パラメータ

・*lpTopLevelExceptionFilter*　UnhandledExceptionFilter関数へ制御が移るときに呼ばれるトップレベルの例外フィルタ関数へのポインタ。このパラメータがNULLの場合は、UnhandledExceptionFilter内で扱っているデフォルトを意味する。フィルタ機能は以下の値のうちの1つを戻す。

値	意味
EXCEPTION_EXECUTE_HANDLER	UnhandledExceptionFilterから戻り、関連した例外を実行する。これは通常プロセスの終了を意味する。
EXCEPTION_CONTINUE_EXECUTION	UnhandledExceptionFilterから戻り、例外のポイントから実行を続ける。
EXCEPTION_CONTINUE_SEARCH	UnhandledExceptionFilterの正常な実行を続行する。

戻り値

SetUnhandledExceptionFilter関数は、設定された以前の例外フィルタのアドレスを戻す。戻り値がNULLの場合は、現在のトップレベルの例外がまったくないことを意味している。

解　説

SetUnhandledExceptionFilterは、呼び出しプロセスでのすべてのスレッドのトップレベル例外フィルタを設定する。

参　照

→UnhandledExceptionFilter

UnhandledExceptionFilter	kernel32.dll	95	NT
アプリケーションの例外をフィルタする			

V　C

```
LONG UnhandledExceptionFilter(
    STRUCT EXCEPTION_POINTERS     *ExceptionInfo
                                  //EXCEPTION_POINTERS構造体へのポインタ
    );
```

V　B

Declare Function UnhandledExceptionFilter Lib "kernel32" Alias "UnhandledException Filter" (ExceptionInfo As EXCEPTION_POINTERS) As Long

パラメータ

・*ExceptionInfo*　　例外時の例外記述とプロセッサステートメントを含むEXCEPTION_POINTERS構造体へのポインタを指定する。このポインタはGetExceptionInformation関数の戻り値である。

戻り値

次の値のいずれかを返す。

値	意味
EXCEPTION_CONTINUE_SEARCH	プロセスからバグを取り除き、例外はアプリケーションのデバッガに渡される。
EXCEPTION_EXECUTE_HANDLER	以前の呼び出しでSEM_NOGPFAULTERRORBOXフラグがSetErrorModeに指定されていた場合、アプリケーションはエラーメッセージボックスを表示しない。

解　説

プロセスからバグを取り除けない場合、関数はアプリケーションのエラーメッセージボックスを表示する。デフォルトではダイアログボックスを表示する必要があるが、呼び出し時にSetErrorMode関数でSEM_NOGPFAULTERRORBOXを指定することで無効にできる。システムは、内部でのプロセスとスレッドの作成時に起こるハンドル例外にUnhandledExceptionFilterを使う。

参　照

→EXCEPTION_POINTERS,→GetExceptionInformation,SetErrorMode,
→SetUnhandledExceptionFilter

System Information

GetComputerName

GetComputerName	kernel32.dll	95	NT

現在のコンピュータ名を取得

V C

```
BOOL GetComputerName(
    LPTSTR      lpBuffer,   //コンピュータ名を格納する文字列へのポインタ
    LPDWORD     nSize       //コンピュータ名を格納するバッファサイズ
    );
```

V B

```
Declare Function GetComputerName Lib "kernel32" Alias "GetComputerNameA"
(ByVal lpBuffer As String, nSize As Long) As Long
```

パラメータ

- *lpBuffer*　コンピュータ名を受け取るNULLで終わる文字列を格納するバッファへのポインタを指定する。
- *nSize*　*lpBuffer*パラメータのサイズをバイト数で指定する。ここで指定するバイト数はコンピュータ名と終端のNULL文字の合計値である。最大値はMAX_COMPUTERNAME_LENGTH＋1バイトで十分である。

戻り値

正常終了	TRUE。*nSize*パラメータにNULL文字をのぞいた実サイズが返される
異常終了	FALSE

拡張エラー情報はGetLastError関数で取得。

参　照

SetComputerName

GetSystemInfo

GetSystemInfo	kernel32.dll	95	NT

現在のシステム情報を返す

V C

```
void GetSystemInfo(
    LPSYSTEM_INFO    lpSystemInfo    //SYSTEM_INFO構造体へのポインタ
    );
```

V B

```
Declare Sub GetSystemInfo Lib "kernel32" Alias "GetSystemInfo" (lpSystemInfo As
SYSTEM_INFO)
```

パラメータ

- *lpSystemInfo*　SYSTEM_INFO構造体へのポインタを指定する。

参　照

→SYSTEM_INFO

GetUserName	advapi32.dll	95	NT
ユーザー名を返す			

V　C

```
BOOL GetUserName(
    LPTSTR    lpBuffer, //ユーザー名を受け取るNULLで終わる文字列へのポインタ
    LPDWORD   nSize     //受け取った文字列を格納するバッファサイズ
);
```

V　B

```
Declare Function GetUserName Lib "advapi32.dll" Alias "GetUserNameA" (ByVal lpBuffer As String, nSize As Long) As Long
```

パラメータ

- *lpBuffer*　ユーザー名を受け取り格納するためのNULLで終わる文字列へのポインタを指定する。パラメータで指定されるバッファが十分なサイズを有していない場合に関数は異常終了する。
- *nSize*　ユーザー名を格納するためのバッファサイズをバイト数で指定する。

戻り値

正常終了	TRUE
異常終了	FALSE

拡張エラー情報はGetLastError関数で取得。

解　説

関数が正常に終了した場合には、*nSize*パラメータに取得したユーザー名のサイズが設定される。設定されるサイズは*lpBuffer*パラメータの内容と異なりNULLで終わる文字列は含まれない。純粋なユーザー名部分のサイズである。

参　照

LookupAccountName

GetVersion	kernel32.dll	95	NT

Windowsのバージョン情報を取得

VC　DWORD GetVersion();

VB　Declare Function GetVersion Lib "kernel32" Alias "GetVersion" () As Long

戻り値　関数が正常に終了した場合は、プラットフォームOSのメジャーバージョン番号とマイナーバージョン番号を返す。

プラットフォーム	上位ワード	下位ワード（メジャーバージョン）
WindowsNT	0	3もしくは4
Windows 95	1	4

解説　関数が正常に終了した場合は、戻り値の下位ワードにWindowsのバージョンが格納される。上位ワードにはマイナーバージョン（リビジョン）番号が2桁の10進数として格納される。

参照　GetVersionEx

GetWindowsDirectory	kernel32.dll	95	NT
Windowsディレクトリのパスを取得			

V　C

```
UINT GetWindowsDirectory(
    LPTSTR  lpBuffer,  //Windowsディレクトリ名を格納するNULLで終わる文字
                         列へのポインタ
    UINT    uSize      //ディレクトリ名を受け取るバッファサイズ
    );
```

V　B

```
Declare Function GetWindowsDirectory Lib "kernel32" Alias "GetWindowsDirectoryA"
(ByVal lpBuffer As String, ByVal nSize As Long) As Long
```

パラメータ

- *lpBuffer*　Windowsディレクトリ名を格納するNULLで終わる文字列へのポインタを指定する。
- *uSize*　Windowsディレクトリ名を受け取るバッファのサイズをバイト数で指定する。

戻り値

正常終了	Windowsディレクトリのパス名のバイト数
異常終了	0

拡張エラー情報はGetLastError関数で取得。

解　説

取得するWindowsディレクトリのパス名には、末尾に\記号は付かない。ただし、Windowsディレクトリがルートディレクトリの場合に限り\記号が付加される。たとえば、CドライブのWindowsというディレクトリにWindowsがインストールされている場合に関数が返す値はC:\Windows。Cドライブのルートにインストールされていた場合にはC:\と返される。

参　照

GetCurrentDirectory,GetSystemDirectory

Time

CompareFileTime	kernel32.dll	95	NT
2つの64ビットのファイル時間を比較			

V C

```
LONG CompareFileTime(
  CONST FILETIME   *lpFileTime1,//日付の比較を行う対象となる第1日付
  CONST FILETIME   *lpFileTime2 //日付の比較を行う対象となる第2日付
  );
```

V B

```
Declare Function CompareFileTime Lib "kernel32" Alias "CompareFileTime" (lpFileTime1 As FILETIME, lpFileTime2 As FILETIME) As Long
```

パラメータ

- *lpFileTime1*　日付の比較を行う第1の64ビット日付を格納するNULLで終わる文字列へのポインタを指定する。
- *lpFileTime2*　日付の比較を行う第2の64ビット日付を格納するNULLで終わる文字列へのポインタを指定する。

戻り値

関数が正常に終了した場合には次の値のいずれかが返される。

-1	第1パラメータの日付が小さい
0	ふたつの日付は等しい
+1	第1パラメータの日付が大きい

拡張エラー情報はGetLastError関数で取得。

参照

GetFileTime,→FILETIME

FileTimeToSystemTime	kernel32.dll	95	NT
64ビット形式の時間をシステム時間に変換			

V　C

```
BOOL FileTimeToSystemTime(
    CONST FILETIME    *lpFileTime,    //変換する64ビット日付へのポインタ
    LPSYSTEMTIME      lpSystemTime    //変換後のシステム日付を格納する
                                        SYSTEMTIME構造体へのポインタ
    );
```

V　B

Declare Function FileTimeToSystemTime Lib "kernel32" Alias "FileTime ToSystemTime" (lpFileTime As FILETIME, lpSystemTime As SYSTEMTIME) As Long

パラメータ

- *lpFileTime*　日付形式の変換を行う64ビット日付へのポインタを指定する。
- *lpSystemTime*　変換された日付を格納するためのSYSTEMTIME構造体へのポインタを指定する。

戻り値

正常終了	TRUE
異常終了	FALSE

拡張エラー情報はGetLastError関数で取得。

解　説

日付の変換に失敗した場合に関数はFILETIME構造体に0x8000000000000000以上の値を返す。

参　照

DosDateTimeToFileTime,→FILETIME,FileTimeToDosDateTime,→SYSTEMTIME, SystemTimeToFileTime

GetSystemTime	kernel32.dll	95	NT
システムの時刻と日付を返す			

V　C

```
void GetSystemTime(
        LPSYSTEMTIME    lpSystemTime    //SYSTEMTIME構造体へのポインタ
        );
```

V　B

Declare Sub GetSystemTime Lib "kernel32" Alias "GetSystemTime" (lpSystemTime As SYSTEMTIME)

パラメータ

- *lpSystemTime*　SYSTEMTIME構造体へのポインタを指定する。

参　照

GetLocalTime,GetSystemTimeAdjustment,→SetSystemTime,→SYSTEMTIME

Window Station and Desktop

CreateDesktop	user32.dll	95	NT
デスクトップを作成			

V C

```
HDESK CreateDesktop(
    LPCTSTR      lpszDesktop,      //デスクトップ名を格納する文字列への
                                     ポインタ
    LPCTSTR      lpszDevice,       //デスクトップを表示する表示装置デバ
                                     イスへのポインタ
    LPDEVMODE    pDevMode,         //予約済み。NULL
    DWORD        dwFlags,          //アプリケーションの関係を示すフラグ
    DWORD        dwDesiredAccess,  //戻されるデスクトップハンドルへのア
                                     クセス
    LPSECURITY_ATTRIBUTES          lpsa    //SECURITY_ATTRIBUTES構
                                             造体へのポインタ
    );
```

V B

```
Declare Function CreateDesktop Lib "user32" Alias "CreateDesktopA" (ByVal lpszDesktop As String, ByVal lpszDevice As String, pDevmode As DEVMODE, ByVal dwFlags As Long, ByVal dwDesiredAccess As Long, lpsa As SECURITY_ATTRIBUTES) As Long
```

パラメータ

- *lpszDesktop*　作成されるデスクトップを指定するNULLで終わる文字列へのポインタ。デスクトップ名に大文字小文字の区別はない。
- *lpszDevice*　デスクトップに割り当てるための表示装置デバイスを識別するNULLで終わる文字列へのポインタ。*lpszDevice*がNULLの場合、ブート時にロードされたデフォルトのディスプレイドライバが使われる。
- *pDevMode*　予約された値。NULLを指定する。
- *dwFlags*　呼び出し元アプリケーションがデスクトップ上の他のアプリケーションとどのように関係するかを示すフラグ。0または以下の値を指定する。

値	意味
DF_ALLOWOTHERACCOUNTHOOK	他のアカウントからの使用も認める

- *dwDesiredAccess*　デスクトップへのアクセスタイプを次のいずれかで指定する。

値	意味
DESKTOP_CREATEMENU	メニューを作成する

DESKTOP_CREATEWINDOW	ウィンドウを作成する
DESKTOP_ENUMERATE	整列させる
DESKTOP_HOOKCONTROL	ウィンドウフック制御を行う
DESKTOP_JOURNALPLAYBACK	ジャーナル記録再生を実行する
DESKTOP_JOURNALRECORD	記録中のジャーナルを実行する
DESKTOP_READOBJECTS	オブジェクトを取得する
DESKTOP_SWITCHDESKTOP	SwitchDesktop関数を使ってデスクトップを実行する
DESKTOP_WRITEOBJECTS	デスクトップにオブジェクトを作成する

・*lpsa*　デスクトップのセキュリティ情報を格納したSECURITY_ATTRIBUTES構造体へのポインタを指定する。

戻り値

正常終了	デスクトップのハンドル
異常終了	NULL

拡張エラー情報はGetLastError関数で取得する。

解　説　CreateDesktop関数はデスクトップにアクセスするために利用するハンドルを返す。

参　照　→SECURITY_ATTRIBUTES,SetProcessWindowStation,→SwitchDesktop

CreateWindowStation　—　NT
ウィンドウステーションの作成

V　C

```
HWINSTA CreateWindowStation(
    LPTSTR                  lpwinsta,        //ウィンドウステーションを示すポインタ
    DWORD                   dwReserved,      //予約済み。NULL
    DWORD                   dwDesiredAccess, //戻されるハンドルへのアクセス
    LPSECURITY_ATTRIBUTES   lpsa,            //SECURITY_ATTRIBUTES構造体へのポインタ
);
```

パラメータ

・*lpwinsta*　作成されるウィンドウステーションの名前を指定するNULLで終わる文字列へのポインタ。ウィンドウステーション名では大文字と小文字は区別されない。管理者グループのメンバーだけに、名前を指定することが許される。*lpwinsta*がNULLの場合、システムはウィンドウステーション名を成形し、呼び出しプロセスのログオンセッション識別子を使う。この名前を得るためには、GetUserObjectInformation関数を利用する。

・*dwReserved*　予約された値。NULLを指定する。

・*dwDesiredAccess*　ウィンドウステーションへのアクセスのタイプを指定する。パラメータは次の値から複数を指定することができる。

値	意味
WINSTA_ACCESSCLIPBOARD	クリップボードを使用する
WINSTA_ACCESSGLOBALATOMS	グローバルアトムを操作する
WINSTA_CREATEDESKTOP	新規のデスクトップオブジェクトを作成する
WINSTA_ENUMDESKTOPS	デスクトップオブジェクトを列挙する
WINSTA_ENUMERATE	ウィンドウステーションを列挙する
WINSTA_EXITWINDOWS	ExitWindowsまたはExitWindowsEx関数の呼び出しを行う
WINSTA_READATTRIBUTES	ウィンドウステーションオブジェクトの属性を取得する
WINSTA_READSCREEN	画面コントロールを操作する
WINSTA_WRITEATTRIBUTES	ウィンドウステーションの属性を変更する

・*lpsa*　ウィンドウステーションのセキュリティ情報を格納したSECURITY_ATTRIBUTES構造体へのポインタを指定する。

戻り値

正常終了	作成されたウィンドウステーションのハンドル
異常終了	NULL

拡張エラー情報はGetLastError関数で取得する

参　照

GetUserObjectInformation,OpenWindowStation

GetProcessWindowStation	user32.dll		NT
ウィンドウステーションハンドルの取得			

V　C

HWINSTA GetProcessWindowStation();

V　B

Declare Function GetProcessWindowStation Lib "user32" Alias "GetProcessWindow Station" () As Long

戻り値

正常終了	呼び出し元プロセスが所属するウィンドウステーションハンドル
異常終了	NULL

拡張エラー情報はGetLastError関数で取得する

解　説

プロセスが発生する場合、システムはウィンドウステーションをプロセスと結び付ける。そのウィンドウステーションを変更するために、プロセスがSetProcessWindowStation関数を利用できる。呼び出しプロセスは、呼び出しにおいて、GetUserObjectInformation、GetUserObjectSecurity、SetUserObjectInformation、およびSetUserObjectSecurity関数から得たハンドルが利用できる。

参　照

GetThreadDesktop,GetUserObjectInformation,GetUserObjectSecurity,
SetProcessWindowStation,SetUserObjectInformation,SetUserObjectSecurity

SwitchDesktop	user32.dll		NT
デスクトップを変更する			

V　C

```
BOOL SwitchDesktop(
    HDESK    hDesktop  //デスクトップを識別するハンドル
    );
```

V　B

Declare Function SwitchDesktop Lib "user32" Alias "SwitchDesktop" (ByVal hDesktop As Long) As Boolean

パラメータ

・*hDesktop*　アクティブなデスクトップを識別するハンドルを指定する。ハンドルはCreateDesktopとOpenDesktop関数の戻り値である。デスクトップが見えないウィンドウステーションに属している場合、SwitchDesktop関数は失敗する。

戻り値

正常終了	TRUE
異常終了	FALSE

拡張エラー情報はGetLastError関数で取得する。

参　照

→CreateDesktop,OpenDesktop

付録

付録A　機能別API一覧

本書の「第2章　Win32 APIリファレンス」で説明しているAPI関数を機能別に分類し、API関数名、機能、掲載ページをまとめた一覧表を以下に掲載する。
アルファベット順の一覧は、「付録B　アルファベット順API一覧」を参照してもらいたい。

●Window：フレームウィンドウオブジェクトの作成から破棄にいたるまでの操作を管理する

●Caret：キャレットオブジェクトの作成から破棄にいたるまでの操作を管理する

●Clipboard：クリップボード操作の制御と管理を目的にした関数

●Common Dialog Box：Windowsが提供するコモンダイアログボックスオブジェクトの作成とデータ通信を中心とした関数

●Cursor：マウスのカーソル形状を管理する関数

CreateCursor	2つのビットマスクからマウスカーソルを作成する	48
GetCursorPos	マウスカーソルの位置を(スクリーン座標で)取得する	49
LoadCursor	マウスカーソルリソースをロードする	49
LoadCursorFromFile	ファイルからマウスカーソルリソースをロードする	51
ShowCursor	マウスカーソル表示カウントを1つ増加または1つ減少させる	52

●Icon：アイコン操作に関する関数

ArrangeIconicWindows	アイコン化されたウィンドウを整列する	52
CreateIconFromResource	アイコンリソースからアイコンを作成する	53

●Message and Message Queue：メッセージ処理を行う関数

DispatchMessage	スクリーンセーバーのパスワードを変更する	54
GetMessage	アプリケーションのメッセージキューからメッセージを取得する	55
SendMessage	1つ以上のウィンドウにメッセージを送る	56

●Mouse Input：マウス入力時の動作と管理を行う関数

ReleaseCapture	マウスキャプチャーを終了する	57
SetCapture	マウスキャプチャーを開始する	57
SetDoubleClickTime	マウスのダブルクリック時間を設定する	58

●Painting and Drawing：フレームウィンドウに対する描画処理を中心とした関数

BeginPaint	描画処理のためにウィンドウを準備する	58
DrawAnimatedRects	アニメーション描画領域を作成する	59
RedrawWindow	ウィンドウを再描画する	60
UpdateWindow	ウィンドウを更新する	62

●Rectangle：指定長方形領域の操作に関する関数

IntersectRect	再描画のための長方形をマークする	62
OffsetRect	指定された長方形を移動する	63
SetRect	長方形の座標を設定する	63

●Timers：タイマーオブジェクトの作成と管理に関する関数

KillTimer	指定されたタイマーイベントを削除する	64
SetTimer	システムタイマーを生成する	65

●Window Class：ウィンドウクラスの管理と操作に関する関数

●Window Property：ウィンドウの属性に関する関数

●Common Controls：Windowsが提供するコモンコントロール操作に関する関数

●Drag List Box：ドラッグリストボックスの操作に関する関数

●Property Sheet：プロパティシートとそのページ操作に関する関数

●Status Window：ステータスバーウィンドウに関する関数

●Toolbar：ツールバーオブジェクトに関する関数

●Up-Down Control：アップダウンコントロールに関する関数

CreateUpDownControl	アップダウンコントロール作成	84

●Button：ボタンコントロールの操作と管理に関する関数

CheckDlgButton	チェックマークを付けるかまたは取り除く、3ステートボタンのときはその状態を変更する	85
CheckRadioButton	指定されたオプションボタンを選択して、そのほかをすべてのボタンを選択解除する	86
IsDlgButtonChecked	ボタンが選択されているかどうかを調べる	87

●Combo Box：コンボボックスコントロールに関する関数

DlgDirListComboBox	パス名と一致するファイルの名前をコンボボックス内に一覧表示する	88
DlgDirSelect	リストボックスから現在の選択項目を文字列バッファにコピーする	90
DlgDirSelectComboBox	コンボボックスから現在の選択項目を文字列バッファにコピーする	91

●Dialog Box：ダイアログウィンドウに関する関数。

CreateDialog	モードレスダイアログボックスを作成する	92
DefDlgProc	アプリケーションで定義されたダイアログボックスプロシージャが処理しないメッセージに対して、デフォルト処理を行う	93
DialogBox	モーダルダイアログボックスを作成する	94
DialogProc	モードレスダイアログボックスに送られるメッセージを処理する	95
EndDialog	リソースを解放して、モーダルダイアログボックスに関連付けられているウィンドウを破棄する	96
GetDialogBaseUnits	ダイアログボックスベース単位取得	97
GetDlgCtrlID	コントロールウィンドウの識別子を返す	98
GetDlgItem	指定されたダイアログボックス内のダイアログボックスコントロールのハンドルを取得する	98
GetNextDlgGroupItem	グループ内の次または前の項目のウィンドウハンドルを返す	99
GetNextDlgTabItem	次または前の項目のWS_TABSTOPコントロールを返す	100
IsDialogMessage	指定されたモードレスダイアログボックスのためのメッセージであるかどうかを判断する	101
MapDialogRect	ダイアログボックス単位変換	102
MessageBox	指定されたテキストとタイトル、アイコンを持つメッセージウィンドウを作成する	103
MessageBoxIndirect	メッセージ構造体からメッセージウィンドウを作成する	109
SendDlgItemMessage	ダイアログボックス内のコントロールにメッセージを送る	106
SetDlgItemInt	コントロールのタイトルまたはテキストを整数値で表された文字列に設定する	107
SetDlgItemText	コントロールのタイトルやテキストを設定する	108

●Edit Control：テキストの入行を行うエディットコントロールオブジェクトに関する関数。

EditWordBreakProc	エディットコントロールテキスト編集	110

●List Box：複数の値から一覧を選択するリストボックスコントロールに関する関数。

DlgDirList	リストボックスに、パスと一致するファイルの名前を一覧表示する	111
DlgDirSelectEx	リストボックス選択項目取得	113

●Scroll Bar：スクロールバーの操作と管理に関する関数

EnableScrollBar	スクロールバー状態設定	114
GetScrollPos	スクロールボックスの現在の位置を取得する	115
GetScrollRange	指定されたスクロールバーの、位置の最大値と最小値を取得する	116
ScrollDC	長方形を水平方向または垂直方向にスクロールする	118
ScrollWindow	クライアント領域の内容を移動する	119
SetScrollPos	スクロールボックスの位置を設定する	121
SetScrollRange	スクロールバーの、位置の最大値と最小値を設定する	122
ShowScrollBar	スクロールバーとそのコントロールを表示したり非表示にしたりする	124

●Control Panel：コントロールパネル操作の関数。

CPlApplet	コントロールパネルDLLのメッセージを処理する	125

●File Installation Library：ファイルインストール機能を提供する関数。

GetFileVersionInfo	指定されたファイルに関するバージョン情報を返す	126
GetFileVersionInfoSize	ファイルのバージョン情報のサイズを返す	127
VerFindFile	インストール位置判定	128
VerInstallFile	ファイルインストール	130
VerLanguageName	言語識別子変換	134
VerQueryValue	バージョンリソース内容取得	136

●Screen Saver：スクリーンセーバー操作と管理に関する関数

DefScreenSaverProc	デフォルトのスクリーンセーバーウィンドウプロシージャを呼び出す	137
RegisterDialogClasses	スクリーンセーバーのダイアログボックスのクラスを登録する	139
ScreenSaverConfigureDialog	設定用ダイアログボックスのメッセージを処理する	140
ScreenSaverProc	スクリーンセーバーのウィンドウメッセージを処理する	141

●Shell Library：シェル上でのユーザーの操作をアプリケーションに伝える

DragAcceptFiles	ウィンドウがドロップされるファイルをウィンドウが受け取れるかどうかを示す情報を登録する	142
DragFinish	ドロップファイル用に割り当てられたメモリを解放する	142
DragQueryFile	ドロップファイルの名前を取得する	143
DragQueryPoint	ファイルがドロップされたときのマウスの位置を取得する	144
ExtractAssociatedIcon	ファイル、または関連付けられた実行可能ファイルから、アイコンハンドルを取得	145
ExtractIcon	実行可能ファイルからアイコンのハンドルを取得	146
FindExecutable	プログラムファイルのためのファイル名とそのハンドルを取得	147
ShellAbout	Windowsが提供するバージョン情報のダイアログボックスを表示	148
ShellExecute	指定されたファイルをオープンまたは表示	149

●Bitmap：Windows標準形式のビットマップオブジェクト操作に関する関数

BitBlt	デバイスコンテキスト間でビットマップをコピー	153
CreateBitmap	デバイスに依存するメモリビットマップを作成	155
CreateCompatibleBitmap	指定されたデバイスと互換性のあるビットマップを作成する	156
CreateDIBitmap	DIB仕様のビットマップからビットマップのハンドルを作成する	157
ExtFloodFill	現在のブラシで領域を塗りつぶす	159
FloodFill	現在のブラシで領域を塗りつぶす	160
GetBitmapBits	指定されたビットマップのメモリ内のビットを取得する	161
GetBitmapDimensionEx	ビットマップの幅と高さを取得	162
GetDIBColorTable	RGBカラー値の回復	163
GetDIBits	CopiesDIBのビットをバッファにコピーする	164
GetPixel	指定されたピクセルのRGBカラー値を取得	166
GetStretchBltMode	現在の伸縮モードを取得する	167
LoadBitmap	ビットマップリソースをロードする	167
SetBitmapBits	ビット配列からビットマップビットを設定	169
SetBitmapDimensionEx	ビットマップの幅と高さを設定	170
SetDIBColorTable	ビットマップからカラー値設定	171
SetDIBits	ビットマップのビットを設定する	172
SetDIBitsToDevice	デバイスにDIBのビットを設定する	173
SetPixel	ピクセルのRGB値を設定する	175
SetPixelV	ピクセルを指定された色に設定	176
SetStretchBltMode	伸縮モードを設定する	177
StretchBlt	コピー元からコピー先のデバイスにビットマップを拡大･縮小コピーする	178
StretchDIBits	転送元から転送先の長方形にDIBを転送する	180

●Brush：グラフィックを塗りつぶす際に利用するブラシオブジェクトに関する関数

CreateBrushIndirect	BITMAP構造体を使ってビットマップを表示	182
CreateDIBPatternBrush	デバイスに依存しないビットマップ(DIB:Device-IndependentBitmap)により定義されるパターンの論理ブラシを作成する	183
CreateDIBPatternBrushPt	ビットマップから論理ブラシを作成	184
CreateHatchBrush	ハッチパターンの論理ブラシを作成する	185
CreatePatternBrush	メモリビットマップ(DIB)により定義されるパターンの論理ブラシを作成する	186
CreateSolidBrush	指定された色を持つ純色のブラシを作成	187
GetBrushOrgEx	現在のブラシの原点を取得	188
GetSysColorBrush	論理ブラシハンドル取得	189
SetBrushOrgEx	現在のブラシの原点を設定	189

●Clipping：クリッピング操作に関する関数

ExcludeClipRect	クリッピングリージョンを変更して長方形を取り除く	191
ExtSelectClipRgn	リージョンをクリッピングリージョンとして選択	192
GetClipBox	クリッピングリージョンを囲む長方形を取得	193
GetClipRgn	現在のクリッピングリージョンを返す	194
GetMetaRgn	現在のメタリージョンを返す	195
IntersectClipRect	クリッピングリージョンと長方形との交わりを作成する	196
OffsetClipRgn	クリッピングリージョンを移動する	197
PtVisible	指定された点が指定されたリージョン内にあるかどうかを調べる	198
RectVisible	長方形の一部が指定されたリージョン内にあるかどうかを調べる	198
SelectClipPath	現在のパスをクリッピングリージョンとして選択	199
SelectClipRgn	クリッピングリージョンを選択する	200
SetMetaRgn	クリッピングリージョンをメタリージョンとして選択	201

●Color：ユーザーインターフェイス用に利用される色に関する関数

AnimatePalette	論理パレットのエントリを置き換えるWindowsは、新しいエントリをシステムパレットに即座に割り当てる	202
CreateHalftonePalette	デバイスコンテキストのためのハーフトーンパレットを作成	203
CreatePalette	論理カラーパレットを作成	204
GetColorAdjustment	デバイスコンテキストに対する色調整を取得	205
GetNearestColor	利用可能な最も近い色を取得	206
GetNearestPaletteIndex	指定の色に最も近いインデックスを取得	207
GetPaletteEntries	論理パレットからエントリを取得する	208
GetSystemPaletteEntries	指定された範囲のパレットエントリをシステムパレットから取得する	209
GetSystemPaletteUse	アプリケーションがシステムパレット全体にアクセスできるかどうかを判断する	210
RealizePalette	論理パレットのエントリをシステムパレットに割り当てる	211
ResizePalette	指定された論理パレットのサイズを変更する	212
SelectPalette	論理パレットをデバイスコンテキストの中に選択する	213
SetColorAdjustment	デバイスコンテキストに対する色の調整値を設定	214
SetPaletteEntries	論理パレットに新しいパレットエントリを設定する	215
SetSystemPaletteUse	アプリケーションがシステムパレット全体にアクセスできるようにする	216
UnrealizeObject	指定されたブラシの原点をリセットするようにGDIに指示する	218
UpdateColors	クライアント領域内の色を更新	219

●Coordinate Space and Transformation：グラフィックの座標系に関する各種の操作と管理を行う関数

●Device Context：デバイスコンテキストに関する関数

ChangeDisplaySettings	ディスプレイの設定を変更する	249
CreateDC	デバイスコンテキストを作成する	251
DeleteDC	デバイスコンテキストを削除する	252
GetDC	クライアント領域のデバイスコンテキストを取得する	253

●Filled Shape：塗りつぶされた図形描画に関する関数

Ellipse	だ円を描画する	254
FillRect	指定のブラシを使用して、指定された長方形を塗りつぶす	255
Pie	扇形を描画する	256
Polygon	多角形を描画する	257

●Font and Text：テキスト出力操作とフォント操作に関する関数

DrawText	書式化されたテキストを長方形内に描画する	258
GetAspectRatioFilterEx	現在のアスペクト比フィルタを取得する	260
GetFontData	TrueTypeフォントファイルから、フォントメトリックデータを取得する	261
PolyTextOut	複数の文字列を描く	262
SetTextAlign	テキスト文字列をデバイスやディスプレイに配置する	263
SetTextColor	テキストの色を設定する	264
TextOut	現在のフォントを使用して、文字列を書き込む	265

●Metafile：メタファイルグラフィックスに関する関数

CreateEnhMetaFile	拡張メタファイルDCを作成	266
CreateMetaFile	メタファイルのディスプレイコンテキストを作成する	267
GetEnhMetaFile	拡張メタファイルを作成	268
GetMetaFile	Windowsメタファイルを作成	269

●Path：パスに関する関数

BeginPath	パスのブラケットを開始	270
CloseFigure	パス中の図形をクローズ	271
EndPath	パスのブラケットを終了	271
GetPath	パス内のすべての直線と曲線を返す	272
PathToRegion	パスからリージョンを作成	273

●Region：リージョンとその操作に関する関数

●Accessibility：Windowsシステム情報に関する関数

●Data Decompression Library：圧縮ファイル操作に関する関数

●Error：Windowsシステムから返される各種のエラー処理に関する関数

●File：ファイルの入出力操作に関する関数

●Help：Windowsヘルプの操作と管理に関する関数

●Memory Management：Windowsで利用されるメモリの操作と管理に関する関数

GlobalAlloc	グローバルヒープからメモリを割り当てる	301
HeapAlloc	ヒープからメモリを割り当てる	302
LocalAlloc	ローカルヒープからメモリを割り当てる	303
VirtualAlloc	指定範囲の仮想ページを予約	304
ZeroMemory	メモリブロックの内容をすべて0にする	305

●Pipe：名前付きおよび名前無しパイプ処理に関する関数

ConnectNamedPipe	クライアントが接続するのを待つ	306
CreateNamedPipe	名前付きパイプのインスタンスを作成	306
CreatePipe	名前なしパイプを作成	309
DisconnectNamedPipe	名前付きパイプのサーバー端点との接続を解除	310

●Process and Thread：プロセスとスレッド処理に関する関数

CreateProcess	新しいプロセスオブジェクトおよびスレッドオブジェクトを作成	311
CreateThread	新しいスレッドを作成	314
GetCurrentProcess	現在のプロセスを識別するハンドルを返す	316
GetCurrentThread	現在のスレッドを識別するハンドルを返す	316
WinExec	ほかのアプリケーションを実行する	317

●Registory：レジストリデータベース処理に関する関数。

RegCreateKey	キーを作成またはオープン	319
RegDeleteKey	キーを削除	320
RegSetValue	指定されたキーにテキスト文字列を関連付ける	321

●Resource：リソース操作に関する関数

LoadImage	リソースからビットマップイメージをロードする	322
LoadResource	リソースをロードする	324

●String Manipulation：文字列操作を実行する関数

CharNext	文字列内の次の文字に移動	324
CharUpper	1文字または文字列を大文字に変換	325
CompareString	2つの文字列を比較	325
FormatMessage	メッセージ文字列の形式を設定	327
GetTimeFormat	時間文字列を取得	329
LoadString	文字列リソースをロードする	331
MultiByteToWideChar	マルチバイト文字列をワイド文字の文字列にマップ	332

●Structured Exception Handling：構造化例外処理に関する関数

AbnormalTermination	try-finallyの状態を返す	334
GetExceptionCode	例外の種類を返す	334
GetExceptionInformation	例外の詳細情報を返す	336
RaiseException	例外を起こす	336
SetUnhandledExceptionFilter	既存のものに代わるフィルタ例外関数を設定	337
UnhandledExceptionFilter	アプリケーションへの例外をフィルタする	338

●System Information：ローカルコンピュータ固有の情報の取得に関する関数

GetComputerName	現在のコンピュータ名を取得	339
GetSystemInfo	現在のシステム情報を返す	339
GetUserName	ユーザー名を返す	340
GetVersion	Windowsのバージョン情報を取得する	341
GetWindowsDirectory	Windowsディレクトリのパスを取得する	342

●Time：日付と時刻に関する関数

CompareFileTime	2つの64ビットのファイル時間を比較	343
FileTimeToSystemTime	64ビット形式の時間をシステム時間に変換	344
GetSystemTime	システムの時刻と日付を返す	344

●Window Station and Desktop：Windowsシステムのデスクトップ操作に関する関数

CreateDesktop	ウィンドウステーションに新規デスクトップを作成する	345
CreateWindowStation	新規のウィンドウステーションを作成する	346
GetProcessWindowStation	プロセスのウィンドウステーションハンドルを返す	348
SwitchDesktop	デスクトップを交換する	348

付録B　アルファベット順API一覧

本書の「第２章　Win32APIリファレンス」で説明しているAPI関数を関数名のアルファベット順に並べ替え、API関数名、機能、掲載ページをまとめた一覧表を以下に掲載する。機能別のAPI一覧は、「付録A　機能別API一覧」を参照してもらいたい。

API関数	概　　要	ページ
AbnormalTermination	try-finallyの状態を返す	334
AddPropSheetPageProc	プロパティシートページ追加	73
AdjustWindowRect	指定されたクライアント領域に合うように、ウィンドウのサイズを計算する	28
AnimatePalette	論理パレットのエントリを置き換えるWindowsは、新しいエントリをシステムパレットに即座に割り当てる	202
ArrangeIconicWindows	アイコン化されたウィンドウを整列する	52
Beep	ビープを発生	284
BeginPaint	描画処理のためにウィンドウを準備する	58
BeginPath	パスのブラケットを開始	270
BitBlt	デバイスコンテキスト間でビットマップをコピー	153
ChangeDisplaySettings	ディスプレイセッティングの変更をする	294
CharNext	文字列内の次の文字に移動	324
CharUpper	1文字または文字列を大文字に変換	325
CheckDlgButton	チェックマークを付けるか、または取り除く3ステートボタンのときは、その状態を変更する	85
CheckRadioButton	指定されたオプションボタンを選択して、そのほかをすべてのボタンを選択解除する	86
ChooseColor	色を選択するためのダイアログボックスを作成する	40
ChooseFont	フォントを選択するためのダイアログボックスを作成する	41
ClientToScreen	クライアント座標をスクリーン座標に変換	220
CloseClipboard	クリップボードをクローズする	38
CloseFigure	パス中の図形をクローズ	271
CombineTransform	2つの座標変換を結合	221
CompareFileTime	2つの64ビットのファイル時間を比較	343
CompareString	2つの文字列を比較	325
ConnectNamedPipe	クライアントが接続するのを待つ	306
CopyFile	ファイルをコピー	287
CPlApplet	コントロールパネルDLLのメッセージを処理する	125
CreateBitmap	デバイスに依存するメモリビットマップを作成	155
CreateBrushIndirect	BITMAP構造体を使ってビットマップを表示	182
CreateCaret	キャレットを作成する	36

API関数	概　要	ページ
CreateCompatibleBitmap	指定されたデバイスと互換性のあるビットマップを作成する	156
CreateCursor	2つのビットマスクからマウスカーソルを作成する	48
CreateDC	デバイスコンテキストを作成する	251
CreateDesktop	ウィンドウステーションに新規デスクトップを作成する	345
CreateDialog	モードレスダイアログボックスを作成する	92
CreateDIBitmap	DIB仕様のビットマップからビットマップのハンドルを作成する	157
CreateDIBPatternBrush	デバイスに依存しないビットマップ(DIB:Device-IndependentBitmap)により定義されるパターンの論理ブラシを作成する	183
CreateDIBPatternBrushPt	ビットマップから論理ブラシを作成	184
CreateEnhMetaFile	拡張メタファイルDCを作成	266
CreateHalftonePalette	デバイスコンテキストのためのハーフトーンパレットを作成	203
CreateHatchBrush	ハッチパターンの論理ブラシを作成する	185
CreateIconFromResource	アイコンリソースからアイコンを作成する	53
CreateMappedBitmap	ツールバービットマップ作成	81
CreateMetaFile	メタファイルのディスプレイコンテキストを作成する	267
CreateNamedPipe	名前付きパイプのインスタンスを作成	306
CreatePalette	論理カラーパレットを作成	204
CreatePatternBrush	メモリビットマップ(DIB)により定義されるパターンの論理ブラシを作成する	186
CreatePipe	名前なしパイプを作成	309
CreateProcess	新しいプロセスオブジェクトおよびスレッドオブジェクトを作成	311
CreatePropertySheetPage	新規プロパティシートページ作成	74
CreateSolidBrush	指定された色を持つ純色のブラシを作成	187
CreateStatusWindow	ステータスウィンドウ作成	78
CreateThread	新しいスレッドを作成	314
CreateToolbarEx	ツールバーボタン作成	82
CreateUpDownControl	アップダウンコントロール作成	84
CreateWindow	オーバラップウィンドウ、ポップアップウィンドウ、子ウィンドウを作成する	29
CreateWindowStation	新規のウィンドウステーションを作成する	346
DefDlgProc	アプリケーションで定義されたダイアログボックスプロシージャが処理しないメッセージに対して、デフォルト処理を行う	93
DefScreenSaverProc	デフォルトのスクリーンセーバーウィンドウプロシージャを呼び出す	137
DeleteDC	デバイスコンテキストを削除する	252
DestroyPropertySheetPage	プロパティシートページ破棄	74
DestroyWindow	ウィンドウを破棄する	32
DialogBox	モーダルダイアログボックスを作成する	94
DialogProc	モードレスダイアログボックスに送られるメッセージを処理する	95
DisconnectNamedPipe	名前付きパイプのサーバー端点との接続を解除	310
DispatchMessage	スクリーンセーバーのパスワードを変更する	54

API関数	概　要	ページ
DlgDirList	リストボックスに、パスと一致するファイルの名前を一覧表示する	111
DlgDirListComboBox	パス名と一致するファイルの名前をコンボボックス内に一覧表示する	88
DlgDirSelect	リストボックスから現在の選択項目を文字列バッファにコピーする	90
DlgDirSelectComboBox	コンボボックスから現在の選択項目を文字列バッファにコピーする	91
DlgDirSelectEx	リストボックス選択項目取得	113
DPtoLP	デバイス座標上の複数の点(ウィンドウの原点からの相対位置)を論理座標点に変換する	222
DragAcceptFiles	ウィンドウがドロップされるファイルをウィンドウが受け取れるかどうかを示す情報を登録する	142
DragFinish	ドロップファイル用に割り当てられたメモリを解放する	142
DragQueryFile	ドロップファイルの名前を取得する	143
DragQueryPoint	ファイルがドロップされたときのマウスの位置を取得する	144
DrawAnimatedRects	アニメーション描画領域を作成する	59
DrawInsert	アイコン描画	72
DrawStatusText	ステータスウィンドウテキスト描画	78
DrawText	書式化されたテキストを長方形内に描画する	258
EditWordBreakProc	エディットコントロールテキスト編集	110
Ellipse	だ円を描画する	254
EmptyClipboard	クリップボードを空にし、クリップボードの所有権を再度割り当てる	39
EnableScrollBar	スクロールバー状態設定	114
EndDialog	リソースを解放して、モーダルダイアログボックスに関連付けられているウィンドウを破棄する	96
EndPath	パスのブラケットを終了	271
ExcludeClipRect	クリッピングリージョンを変更して長方形を取り除く	191
ExitWindows	Windowsをシャットダウンする	285
ExtensionPropSheetPageProc	プロパティシート拡張	75
ExtFloodFill	現在のブラシで領域を塗りつぶす	159
ExtractAssociatedIcon	ファイル、または関連付けられた実行可能ファイルから、アイコンハンドルを取得	145
ExtractIcon	実行可能ファイルからアイコンのハンドルを取得	146
ExtSelectClipRgn	リージョンをクリッピングリージョンとして選択	192
FileTimeToSystemTime	64ビット形式の時間をシステム時間に変換	344
FillRect	指定のブラシを使用して、指定された長方形を塗りつぶす	255
FillRgn	指定されたリージョンをブラシパターンで塗りつぶす	274
FindExecutable	プログラムファイルのためのファイル名とそのハンドルを取得	147
FindText	テキストを検索するためのダイアログボックスを作成する	42
FormatMessage	メッセージ文字列の形式を設定	327
GetAspectRatioFilterEx	現在のアスペクト比フィルタを取得する	260
FloodFill	現在のブラシで領域を塗りつぶす	160

API関数	概　要	ページ
GetBitmapBits	指定されたビットマップのメモリ内のビットを取得する	161
GetBitmapDimensionEx	ビットマップの幅と高さを取得	162
GetBrushOrgEx	現在のブラシの原点を取得	188
GetCaretPos	現在のキャレットの位置を返す	37
GetClassInfo	ウィンドウクラス情報を取得する	66
GetClipboardData	クリップボードからデータを取得する	39
GetClipBox	クリッピングリージョンを囲む長方形を取得	193
GetClipRgn	現在のクリッピングリージョンを返す	194
GetColorAdjustment	デバイスコンテキストに対する色調整を取得	205
GetComputerName	現在のコンピュータ名を取得	339
GetCurrentDirectory	プロセスの現在のディレクトリを返す	288
GetCurrentPositionEx	位置を論理単位で取得	223
GetCurrentProcess	現在のプロセスを識別するハンドルを返す	316
GetCurrentThread	現在のスレッドを識別するハンドルを返す	316
GetCursorPos	マウスカーソルの位置を(スクリーン座標で)取得する	49
GetDC	クライアント領域のデバイスコンテキストを取得する	253
GetDeviceCaps	ディスプレイデバイスの固有情報を取得	223
GetDialogBaseUnits	ダイアログボックスベース単位取得	97
GetDIBColorTable	RGBカラー値の回復	163
GetDIBits	CopiesDIBのビットをバッファにコピーする	164
GetDiskFreeSpace	空きディスク容量の総量を返す	289
GetDlgCtrlID	コントロールウィンドウの識別子を返す	98
GetDlgItem	指定されたダイアログボックス内のダイアログボックスコントロールのハンドルを取得する	98
GetEffectiveClientRect	クライアント領域矩形範囲計算	70
GetEnhMetaFile	拡張メタファイルを作成	268
GetExceptionCode	例外の種類を返す	334
GetExceptionInformation	例外の詳細情報を返す	336
GetFileAttributes	ファイル属性を返す	290
GetFileSize	指定されたファイルのサイズを返す	291
GetFileTitle	ファイル名を取得	43
GetFileType	指定されたファイルの種類を返す	292
GetFileVersionInfo	指定されたファイルに関するバージョン情報を返す	126
GetFileVersionInfoSize	ファイルのバージョン情報のサイズを返す	127
GetFullPathName	ファイル名のパス名を取得	293
GetGraphicsMode	指定されたDCに対するグラフィックスモードを取得	230
GetLastError	拡張エラーコードを返す	286

API関数	概　　要	ページ
GetFontData	TrueTypeフォントファイルから、フォントメトリックデータを取得する	261
GetMapMode	現在のマッピングモードを取得する	231
GetMenuContextHelpId	メニューコンテキストヘルプIDを取得する	298
GetMessage	アプリケーションのメッセージキューからメッセージを取得する	55
GetMetaFile	Windowsメタファイルを作成	269
GetMetaRgn	現在のメタリージョンを返す	195
GetNearestColor	利用可能な最も近い色を取得	206
GetNearestPaletteIndex	指定の色に最も近いインデックスを取得	207
GetNextDlgGroupItem	グループ内の次または前の項目のウィンドウハンドルを返す	99
GetNextDlgTabItem	次または前の項目のWS_TABSTOPコントロールを返す	100
GetOpenFileName	ファイルをオープンするためのダイアログボックスを作成する	43
GetPaletteEntries	論理パレットからエントリを取得する	208
GetPath	パス内のすべての直線と曲線を返す	272
GetPixel	指定されたピクセルのRGBカラー値を取得	166
GetProcessWindowStation	プロセスのウィンドウステーションハンドルを返す	348
GetProp	プロパティリストを取得する	68
GetRgnBox	リージョンの境界長方形の座標を取得する	275
GetSaveFileName	ファイルを保存するためのダイアログボックスを作成する	44
GetScrollPos	スクロールボックスの現在の位置を取得する	115
GetScrollRange	指定されたスクロールバーの、位置の最大値と最小値を取得する	116
GetStretchBltMode	現在の伸縮モードを取得する	167
GetSysColorBrush	論理ブラシハンドル取得	189
GetSystemInfo	現在のシステム情報を返す	339
GetSystemMetrics	システムメトリックに関する情報を取得する	276
GetSystemPaletteEntries	指定された範囲のパレットエントリをシステムパレットから取得する	209
GetSystemPaletteUse	アプリケーションがシステムパレット全体にアクセスできるかどうかを判断する	210
GetSystemTime	システムの時刻と日付を返す	344
GetTimeFormat	時間文字列を取得	329
GetUserName	ユーザー名を返す	340
GetVersion	Windowsのバージョン情報を取得する	341
GetViewportExtEx	ビューポートの範囲を取得	232
GetWindow	ウィンドウハンドルを取得する	33
GetWindowContextHelpId	ウィンドウコンテキストヘルプIDを取得する	298
GetWindowExtEx	ウィンドウの範囲を取得	233
GetWindowsDirectory	Windowsディレクトリのパスを取得する	342
GetWindowText	ウィンドウタイトルをバッファにコピーする	34
GetWorldTransform	ワールド空間からページ空間への変換の取得	234

API関数	概　要	ページ
GetWindowRect	ウィンドウ全体の寸法を取得する	34
GlobalAlloc	グローバルヒープからメモリを割り当てる	301
HeapAlloc	ヒープからメモリを割り当てる	302
HideCaret	指定されたウィンドウからキャレットを削除する	38
InitCommonControls	コモンコントロールDLL確立	71
IntersectClipRect	クリッピングリージョンと長方形との交わりを作成する	196
IntersectRect	再描画のための長方形をマークする	62
IsDialogMessage	指定されたモードレスダイアログボックスのためのメッセージがであるかどうかを判断する	101
IsDlgButtonChecked	ボタンが選択されているかどうかを調べる	87
KillTimer	指定されたタイマーイベントを削除する	64
LBItemFromPt	リスト項目インデックス検出	72
LoadBitmap	ビットマップリソースをロードする	167
LoadCursor	マウスカーソルリソースをロードする	49
LoadCursorFromFile	ファイルからカーソルリソースをロードする	51
LoadImage	リソースからビットマップイメージをロードする	322
LoadResource	リソースをロードする	324
LoadString	文字列リソースをロードする	331
LocalAlloc	ローカルヒープからメモリを割り当てる	303
LockFile	ファイルをロックする	294
LPtoDP	論理座標点をデバイス座標点に変換する	235
LZClose	ファイルをクローズする	280
LZCopy	ファイルをコピーして圧縮されていればそれを展開する	281
LZOpenFile	圧縮および非圧縮のファイルをオープンする	282
MakeDragList	ドラッグリストボックス作成	73
MapDialogRect	ダイアログボックス単位変換	102
MapWindowPoints	座標点をほかの座標系に変換する	236
MenuHelp	ステータスウィンドウヘルプ表示	80
MessageBox	指定されたテキストとタイトル、アイコンを持つメッセージウィンドウを作成する	103
MessageBoxIndirect	メッセージ構造体からメッセージウィンドウを作成する	109
ModifyWorldTransform	現在のワールド座標変換を変更	237
MoveFile	ファイル名を変更	295
MoveWindow	ウィンドウのサイズと位置を変更する	35
MultiByteToWideChar	マルチバイト文字列をワイド文字の文字列にマップ	332
OffsetRect	指定された長方形を移動する	63
OffsetViewportOrgEx	ビューポートの原点を移動	238
OffsetWindowOrgEx	ウィンドウの原点を移動	239

API関数	概　　要	ページ
OffsetClipRgn	クリッピングリージョンを移動する	197
OpenClipboard	クリップボードをオープンする	40
PageSetupDlg	印刷ページ設定のダイアログを作成する	45
PaintRgn	選択されているブラシパターンでリージョンを塗りつぶす	275
PathToRegion	パスからリージョンを作成	273
Pie	扇形を描画する	256
Polygon	多角形を描画する	257
PolyTextOut	複数の文字列を描く	262
PrintDlg	テキストを印刷するためのダイアログボックスを作成する	46
PropertySheet	プロパティシート表示	75
PropSheetPageProc	プロパティシートページ初期化	76
PropSheetProc	プロパティシート初期化	77
PtVisible	指定された点が指定されたリージョン内にあるかどうかを調べる	198
RaiseException	例外を起こす	336
ReadFile	ファイルから読み取る	296
RealizePalette	論理パレットのエントリをシステムパレットに割り当てる	211
RectVisible	長方形の一部が指定されたリージョン内にあるかどうかを調べる	198
RedrawWindow	ウィンドウを再描画する	60
RegCreateKey	キーを作成またはオープン	319
RegDeleteKey	キーを削除	320
RegisterClass	ウィンドウクラスを登録する	66
RegisterDialogClasses	スクリーンセーバーのダイアログボックスのクラスを登録する	139
RegSetValue	指定されたキーにテキスト文字列を関連付ける	321
ReleaseCapture	マウスキャプチャーを終了する	57
RemoveDirectory	既存のディレクトリを除去	297
ReplaceText	テキストを置換するためのダイアログボックスを作成する	47
ResizePalette	指定された論理パレットのサイズを変更する	212
ScaleViewportExtEx	ビューポートの範囲をスケーリング	240
ScaleWindowExtEx	ウィンドウの範囲をスケーリング	241
ScreenSaverConfigureDialog	設定用ダイアログボックスのメッセージを処理する	140
ScreenSaverProc	スクリーンセーバーのウィンドウメッセージを処理する	141
ScreenToClient	指定されたディスプレイ上の点のスクリーン座標をクライアント座標に変換する	242
ScrollDC	長方形を水平方向または垂直方向にスクロールする	118
ScrollWindow	クライアント領域の内容を移動する	119
SelectClipPath	現在のパスをクリップリージョンとして選択	199
SelectClipRgn	クリッピングリージョンを選択する	200

API関数	概　要	ページ
SelectPalette	論理パレットをデバイスコンテキストの中に選択する	213
SendDlgItemMessage	ダイアログボックス内のコントロールにメッセージを送る	106
SendMessage	1つ以上のウィンドウにメッセージを送る	56
SetBitmapBits	ビット配列からビットマップビットを設定	169
SetBitmapDimensionEx	ビットマップの幅と高さを設定	170
SetBrushOrgEx	現在のブラシの原点を設定	189
SetCapture	マウスキャプチャーを開始する	57
SetColorAdjustment	デバイスコンテキストに対する色の調整値を設定	214
SetDIBColorTable	ビットマップからカラー値設定	171
SetDIBits	ビットマップのビットを設定する	172
SetDIBitsToDevice	デバイスにDIBのビットを設定する	173
SetDlgItemInt	コントロールのタイトルまたはテキストを整数値で表された文字列に設定する	107
SetDlgItemText	コントロールのタイトルやテキストを設定する	108
SetDoubleClickTime	マウスのダブルクリック時間を設定する	58
SetGraphicsMode	指定されたDCに対するグラフィックスモードを設定	242
SetMapMode	指定されたデバイスコンテキストのマッピングモードを設定する	244
SetMetaRgn	クリップリージョンをメタリージョンとして選択	201
SetPaletteEntries	論理パレットに新しいパレットエントリを設定する	215
SetPixel	ピクセルのRGB値を設定する	175
SetPixelV	ピクセルを指定された色に設定	176
SetProp	プロパティリストを設定する	68
SetRect	長方形の座標を設定する	63
SetScrollPos	スクロールボックスの位置を設定する	121
SetScrollRange	スクロールバーの、位置の最大値と最小値を設定する	122
SetStretchBltMode	伸縮モードを設定する	177
SetSystemPaletteUse	アプリケーションがシステムパレット全体にアクセスできるようにする	216
SetTextAlign	テキスト文字列をデバイスやディスプレイに配置する	263
SetTextColor	テキストの色を設定する	264
SetTimer	システムタイマを生成する	65
SetUnhandledExceptionFilter	既存のものに代わるフィルタ例外関数を設定	337
SetViewportExtEx	ビューポートの範囲を設定	246
SetWindowExtEx	ウィンドウの範囲を設定	247
SetWindowText	ウィンドウのタイトルやテキストを設定する	36
SetWorldTransform	現在のワールド座標変換を設定	248
ShellAbout	Windowsが提供するバージョン情報のダイアログボックスを表示	148
ShellExecute	指定されたファイルをオープンまたは表示	149
ShowCursor	マウスカーソル表示カウントを1つ増加または1つ減少させる	52

API関数	概　要	ページ
ShowHideMenuCtl	メニュー項目チェック属性と表示非表示設定	71
ShowScrollBar	スクロールバーとそのコントロールを表示したり非表示にしたりする	124
StretchBlt	コピー元からコピー先のデバイスにビットマップを拡大·縮小コピーする	178
StretchDIBits	転送元から転送先の長方形にDIBを転送する	180
SwitchDesktop	デスクトップを交換する	348
TextOut	現在のフォントを使用して、文字列を書き込む	265
UnhandledExceptionFilter	アプリケーションへの例外をフィルタする	338
UnrealizeObject	指定されたブラシの原点をリセットするようにGDIに指示する	218
UnregisterClass	ウィンドウクラスをウィンドウクラステーブルから削除する	67
UpdateColors	クライアント領域内の色を更新	219
UpdateWindow	ウィンドウを更新する	62
VerFindFile	インストール位置判定	128
VerInstallFile	ファイルインストール	130
VerLanguageName	言語識別子変換	134
VerQueryValue	バージョンリソース内容取得	136
VirtualAlloc	指定範囲の仮想ページを予約	304
WinExec	ほかのアプリケーションを実行する	317
WinHelp	Windowsヘルプアプリケーションを起動し、コンテキスト情報やトピック情報を渡す	298
ZeroMemory	メモリブロックの内容をすべて0にする	305

付録C　Visual Basic用構造体定義一覧

以下は、本書で扱っているAPI関数が利用する構造体をアルファベット順に列挙した一覧である。利用方法やコーディングについては「1.5　Visual BasicからのWindows API呼び出し」を参照してもらいたい。Visual C++では、ヘッダファイルをインクルードすることで定義されるので特に利用する必要はない。

```
Type ABC
    abcA As Long
    abcB As Long
    abcC As Long
End Type
```

```
Type ABCFLOAT
    abcfA As Double
    abcfB As Double
    abcfC As Double
End Type
```

```
Type ACCEL
    fVirt As Byte
    key As Integer
    cmd As Integer
End Type
```

```
Type ACCESS_ALLOWED_ACE
    Header As ACE_HEADER
    Mask As Long
    SidStart As Long
End Type
```

```
Type ACCESS_DENIED_ACE
    Header As ACE_HEADER
    Mask As Long
    SidStart As Long
End Type
```

```
Type ACE_HEADER
    AceType As Byte
    AceFlags As Byte
    AceSize As Long
End Type
```

```
Type ACL
    AclRevision As Byte
    Sbz1 As Byte
    AclSize As Integer
    AceCount As Integer
    Sbz2 As Integer
End Type
```

```
Type ACL_REVISION_INFORMATION
    AclRevision As Long
End Type
```

```
Type ACL_SIZE_INFORMATION
    AceCount As Long
    AclBytesInUse As Long
    AclBytesFree As Long
End Type
```

```
Type ACTION_HEADER
    transport_id As Long
    action_code As Integer
    Reserved As Integer
End Type
```

```
Type ADAPTER_STATUS
    adapter_address As String*6
    rev_major As Integer
    reserved0 As Integer
    adapter_type As Integer
    rev_minor As Integer
    duration As Integer
    frmr_recv As Integer
    frmr_xmit As Integer
    iframe_recv_err As Integer
    xmit_aborts As Integer
    xmit_success As Long
    recv_success As Long
    iframe_xmit_err As Integer
    recv_buff_unavail As Integer
    t1_timeouts As Integer
    ti_timeouts As Integer
    Reserved1 As Long
    free_ncbs As Integer
    max_cfg_ncbs As Integer
    max_ncbs As Integer
    xmit_buf_unavail As Integer
    max_dgram_size As Integer
    pending_sess As Integer
    max_cfg_sess As Integer
    max_sess As Integer
    max_sess_pkt_size As Integer
    name_count As Integer
End Type
```

```
Type ADDJOB_INFO_1
    Path As String
    JobId As Long
End Type
```

```
Type ANIMATIONINFO
    cbSize As Long
    iMinAnimate As Long
End Type
```

```
Type APPBARDATA
    cbSize As Long
    hwnd As Long
    uCallbackMessage As Long
    uEdge As Long
    rc As Rect
    lParam As Long
End Type
```

```
Type AUXCAPS
    wMid As Integer
    wPid As Integer
    vDriverVersion As Long
    szPname As
            String*MAXPNAMELEN
    wTechnology As Integer
    dwSupport As Long
End Type
```

```
Type BITMAP
    bmType As Long
    bmWidth As Long
    bmHeight As Long
    bmWidthBytes As Long
    bmPlanes As Integer
    bmBitsPixel As Integer
    bmBits As Long
End Type
```

```
Type BITMAPCOREHEADER
    bcSize As Long
    bcWidth As Integer
    bcHeight As Integer
    bcPlanes As Integer
    bcBitCount As Integer
End Type
```

```
Type BITMAPCOREINFO
    bmciHeader
            As BITMAPCORE HEADER
    bmciColors As RGBTRIPLE
End Type
```

```
Type BITMAPFILEHEADER
    bfType As Integer
    bfSize As Long
    bfReserved1 As Integer
    bfReserved2 As Integer
    bfOffBits As Long
End Type
```

```
Type BITMAPINFO
    bmiHeader
            As BITMAPINFOHEADER
    bmiColors As RGBQUAD
End Type
```

```
Type BITMAPINFOHEADER
    biSize As Long
    biWidth As Long
    biHeight As Long
    biPlanes As Integer
    biBitCount As Integer
    biCompression As Long
    biSizeImage As Long
    biXPelsPerMeter As Long
    biYPelsPerMeter As Long
    biClrUsed As Long
    biClrImportant As Long
End Type
```

```
Type BITMAPV4HEADER
    bV4Size As Long
    bV4Width As Long
    bV4Height As Long
    bV4Planes As Integer
    bV4BitCount As Integer
    bV4V4Compression As Long
    bV4SizeImage As Long
    bV4XPelsPerMeter As Long
    bV4YPelsPerMeter As Long
    bV4ClrUsed As Long
    bV4ClrImportant As Long
    bV4RedMask As Long
    bV4GreenMask As Long
    bV4BlueMask As Long
    bV4AlphaMask As Long
    bV4CSType As Long
    bV4Endpoints As Long
    bV4GammaRed As Long
    bV4GammaGreen As Long
    bV4GammaBlue As Long
End Type
```

```
Type BY_HANDLE_FILE_
                INFORMATION
    dwFileAttributes As Long
    ftCreationTime As FILETIME
    ftLastAccessTime As FILETIME
    ftLastWriteTime As FILETIME
    dwVolumeSerialNumber As Long
    nFileSizeHigh As Long
    nFileSizeLow As Long
    nNumberOfLinks As Long
    nFileIndexHigh As Long
    nFileIndexLow As Long
End Type
```

```
Type CANDIDATEFORM
    dwIndex As Long
    dwStyle As Long
    ptCurrentPos As POINTAPI
    rcArea As Rect
End Type
```

```
Type CANDIDATELIST
    dwSize As Long
    dwStyle As Long
    dwCount As Long
    dwSelection As Long
    dwPageStart As Long
    dwPageSize As Long
    dwOffset(1) As Long
End Type
```

```
Type CBT_CREATEWND
    lpcs As CREATESTRUCT
    hWndInsertAfter As Long
End Type
```

```
Type CBTACTIVATESTRUCT
    fMouse As Boolean
    hWndActive As Long
End Type
```

```
Type CHAR_INFO
    Char As Integer
    Attributes As Integer
End Type
```

```
Type CHARSETINFO
    ciCharset As Long
    ciACP As Long
    fs As FONTSIGNATURE
End Type
```

```
Type CHOOSECOLOR
    lStructSize As Long
    hwndOwner As Long
    hInstance As Long
    rgbResult As Long
    lpCustColors As Long
    flags As Long
    lCustData As Long
    lpfnHook As Long
    lpTemplateName As String
End Type
```

```
Type CHOOSEFONT
    lStructSize As Long
    hwndOwner As Long
    hdc As Long
    lpLogFont As LOGFONT
    iPointSize As Long
    flags As Long
    rgbColors As Long
    lCustData As Long
    lpfnHook As Long
    lpTemplateName As String
    hInstance As Long
    lpszStyle As String
    nFontType As Integer
    MISSING_ALIGNMENT As Integer
    nSizeMin As Long
    nSizeMax As Long
End Type
```

```
Type CIEXYZ
    ciexyzX As Long
    ciexyzY As Long
    ciexyzZ As Long
End Type
```

```
Type CIEXYZTRIPLE
    ciexyzRed As CIEXYZ
    ciexyzGreen As CIEXYZ
    ciexyBlue As CIEXYZ
End Type
```

```
Type CLIENTCREATESTRUCT
    hWindowMenu As Long
    idFirstChild As Long
End Type
```

```
Type COLORADJUSTMENT
    caSize As Integer
    caFlags As Integer
    caIlluminantIndex As Integer
    caRedGamma As Integer
```

```
    caGreenGamma As Integer
    caBlueGamma As Integer
    caReferenceBlack As Integer
    caReferenceWhite As Integer
    caContrast As Integer
    caBrightness As Integer
    caColorfulness As Integer
    caRedGreenTint As Integer
End Type
```

Type **COMMCONFIG**
```
    dwSize As Long
    wVersion As Integer
    wReserved As Integer
    dcbx As DCB
    dwProviderSubType As Long
    dwProviderOffset As Long
    dwProviderSize As Long
    wcProviderData As Byte
End Type
```

Type **COMMPROP**
```
    wPacketLength As Integer
    wPacketVersion As Integer
    dwServiceMask As Long
    dwReserved1 As Long
    dwMaxTxQueue As Long
    dwMaxRxQueue As Long
    dwMaxBaud As Long
    dwProvSubType As Long
    dwProvCapabilities As Long
    dwSettableParams As Long
    dwSettableBaud As Long
    wSettableData As Integer
    wSettableStopParity As Integer
    dwCurrentTxQueue As Long
    dwCurrentRxQueue As Long
    dwProvSpec1 As Long
    dwProvSpec2 As Long
    wcProvChar(1) As Integer
End Type
```

Type **COMMTIMEOUTS**
```
    ReadIntervalTimeout As Long
    ReadTotalTimeoutMultiplier As Long
    ReadTotalTimeoutConstant As Long
    WriteTotalTimeoutMultiplier As Long
    WriteTotalTimeoutConstant As Long
End Type
```

Type **COMPAREITEMSTRUCT**
```
    CtlType As Long
    CtlID As Long
    hwndItem As Long
    itemID1 As Long
    itemData1 As Long
    itemID2 As Long
    itemData2 As Long
End Type
```

Type **COMPOSITIONFORM**
```
    dwStyle As Long
    ptCurrentPos As POINTAPI
    rcArea As Rect
End Type
```

Type **COMSTAT**
```
    fCtsHold As Long
    fDsrHold As Long
    fRlsdHold As Long
    fXoffHold As Long
    fXoffSent As Long
    fEof As Long
    fTxim As Long
    fReserved As Long
    cbInQue As Long
    cbOutQue As Long
End Type
```

Type **CONSOLE_CURSOR_INFO**
```
    dwSize As Long
    bVisible As Boolean
End Type
```

Type **CONSOLE_SCREEN_BUFFER_INFO**
```
    dwSize As COORD
    dwCursorPosition As COORD
    wAttributes As Integer
    srWindow As SMALL_RECT
    dwMaximumWindowSize As COORD
End Type
```

Type **CONTEXT**
```
    FltF0 As Double
    FltF1 As Double
    FltF2 As Double
    FltF3 As Double
    FltF4 As Double
    FltF5 As Double
    FltF6 As Double
    FltF7 As Double
    FltF8 As Double
    FltF9 As Double
    FltF10 As Double
    FltF11 As Double
    FltF12 As Double
    FltF13 As Double
    FltF14 As Double
    FltF15 As Double
    FltF16 As Double
    FltF17 As Double
    FltF18 As Double
    FltF19 As Double
    FltF20 As Double
    FltF21 As Double
    FltF22 As Double
    FltF23 As Double
    FltF24 As Double
    FltF25 As Double
    FltF26 As Double
    FltF27 As Double
    FltF28 As Double
    FltF29 As Double
    FltF30 As Double
    FltF31 As Double
    IntV0 As Double
    IntT0 As Double
    IntT1 As Double
    IntT2 As Double
    IntT3 As Double
    IntT4 As Double
    IntT5 As Double
    IntT6 As Double
    IntT7 As Double
    IntS0 As Double
    IntS1 As Double
    IntS2 As Double
    IntS3 As Double
    IntS4 As Double
    IntS5 As Double
    IntFp As Double
    IntA0 As Double
    IntA1 As Double
    IntA2 As Double
    IntA3 As Double
    IntA4 As Double
    IntA5 As Double
    IntT8 As Double
    IntT9 As Double
    IntT10 As Double
    IntT11 As Double
    IntRa As Double
    IntT12 As Double
    IntAt As Double
    IntGp As Double
    IntSp As Double
    IntZero As Double
    Fpcr As Double
    SoftFpcr As Double
    Fir As Double
    Psr As Long
    ContextFlags As Long
    Fill(4) As Long
End Type
```

```
Type CONVCONTEXT
    cb As Long
    wFlags As Long
    wCountryID As Long
    iCodePage As Long
    dwLangID As Long
    dwSecurity As Long
    qos As SECURITY_QUALITY_
                    OF_SERVICE
End Type
```

```
Type CONVINFO
    cb As Long
    hUser As Long
    hConvPartner As Long
    hszSvcPartner As Long
    hszServiceReq As Long
    hszTopic As Long
    hszItem As Long
    wFmt As Long
    wType As Long
    wStatus As Long
    wConvst As Long
    wLastError As Long
    hConvList As Long
    ConvCtxt As CONVCONTEXT
    hwnd As Long
    hwndPartner As Long
End Type
```

```
Type COORD
    x As Integer
    y As Integer
End Type
```

```
Type COPYDATASTRUCT
    dwData As Long
    cbData As Long
    lpData As Long
End Type
```

```
Type CPINFO
    MaxCharSize As Long
    DefaultChar(MAX_DEFAUL
                TCHAR) As Byte
    LeadByte(MAX_LEADBYTES)
                    As Byte
End Type
```

```
Type CREATE_PROCESS
            _DEBUG_INFO
    hFile As Long
    hProcess As Long
    hThread As Long
    lpBaseOfImage As Long
    dwDebugInfoFileOffset As Long
    nDebugInfoSize As Long
    lpThreadLocalBase As Long
    lpStartAddress As Long
    lpImageName As Long
    fUnicode As Integer
End Type
```

```
Type
    CREATE_THREAD_DEBUG_INFO
    hThread As Long
    lpThreadLocalBase As Long
    lpStartAddress As Long
End Type
```

```
Type CREATESTRUCT
    lpCreateParams As Long
    hInstance As Long
    hMenu As Long
    hWndParent As Long
    cy As Long
    cx As Long
    y As Long
    x As Long
    style As Long
    lpszName As String
    lpszClass As String
    ExStyle As Long
End Type
```

```
Type CRGB
    bRed As Byte
    bGreen As Byte
    bBlue As Byte
    bExtra As Byte
End Type
```

```
Type CRITICAL_SECTION
    dummy As Long
End Type
```

```
Type CURRENCYFMT
    NumDigits As Long
    LeadingZero As Long
    Grouping As Long
    lpDecimalSep As String
    lpThousandSep As String
    NegativeOrder As Long
    PositiveOrder As Long
    lpCurrencySymbol As String
End Type
```

```
Type CWPSTRUCT
    lParam As Long
    wParam As Long
    message As Long
    hwnd As Long
End Type
```

```
Type DATATYPES_INFO_1
    pName As String
End Type
```

```
Type DCB
    DCBlength As Long
    BaudRate As Long
    fBinary As Long
    fParity As Long
    fOutxCtsFlow As Long
    fOutxDsrFlow As Long
    fDtrControl As Long
    fDsrSensitivity As Long
    fTXContinueOnXoff As Long
    fOutX As Long
    fInX As Long
    fErrorChar As Long
    fNull As Long
    fRtsControl As Long
    fAbortOnError As Long
    fDummy2 As Long
    wReserved As Integer
    XonLim As Integer
    XoffLim As Integer
    ByteSize As Byte
    Parity As Byte
    StopBits As Byte
    XonChar As Byte
    XoffChar As Byte
    ErrorChar As Byte
    EofChar As Byte
    EvtChar As Byte
End Type
```

```
Type DDEACK
    bAppReturnCode As Integer
    Reserved As Integer
    fbusy As Integer
    fack As Integer
End Type
```

```
Type DDEADVISE
    Reserved As Integer
    fDeferUpd As Integer
    fAckReq As Integer
    cfFormat As Integer
End Type
```

```
Type DDEDATA
    unused As Integer
    fresponse As Integer
    fRelease As Integer
    Reserved As Integer
    fAckReq As Integer
    cfFormat As Integer
    Value(1) As Byte
End Type
```

```
Type DDELN
    unused As Integer
    fRelease As Integer
    fDeferUpd As Integer
    fAckReq As Integer
    cfFormat As Integer
End Type
```

```
Type DDEML_MSG_HOOK_DATA
    uiLo As Long
    uiHi As Long
    cbData As Long
    Data(8) As Long
End Type
```

```
Type DDEPOKE
    unused As Integer
    fRelease As Integer
    fReserved As Integer
    cfFormat As Integer
    Value(1) As Byte
End Type
```

```
Type DDEUP
    unused As Integer
    fAck As Integer
    fRelease As Integer
    fReserved As Integer
    fAckReq As Integer
    cfFormat As Integer
    rgb(1) As Byte
End Type
```

```
Type DEBUGHOOKINFO
    hModuleHook As Long
    Reserved As Long
    lParam As Long
    wParam As Long
    code As Long
End Type
```

```
Type DELETEITEMSTRUCT
    CtlType As Long
    CtlID As Long
    itemID As Long
    hwndItem As Long
    itemData As Long
End Type
```

```
Type DEVMODE
    dmDeviceName As
            String*CCHDEVICENAME
    dmSpecVersion As Integer
    dmDriverVersion As Integer
    dmSize As Integer
    dmDriverExtra As Integer
    dmFields As Long
    dmOrientation As Integer
    dmPaperSize As Integer
    dmPaperLength As Integer
    dmPaperWidth As Integer
    dmScale As Integer
    dmCopies As Integer
    dmDefaultSource As Integer
    dmPrintQuality As Integer
    dmColor As Integer
    dmDuplex As Integer
    dmYResolution As Integer
    dmTTOption As Integer
    dmCollate As Integer
    dmFormName
            As String*CCHFORMNAME
    dmUnusedPadding As Integer
    dmBitsPerPel As Integer
    dmPelsWidth As Long
    dmPelsHeight As Long
    dmDisplayFlags As Long
    dmDisplayFrequency As Long
End Type
```

```
Type DEVNAMES
    wDriverOffset As Integer
    wDeviceOffset As Integer
    wOutputOffset As Integer
    wDefault As Integer
End Type
```

```
Type DLGITEMTEMPLATE
    style As Long
    dwExtendedStyle As Long
    x As Integer
    y As Integer
    cx As Integer
    cy As Integer
    id As Integer
End Type
```

```
Type DLGTEMPLATE
    style As Long
    dwExtendedStyle As Long
    cdit As Integer
    x As Integer
    y As Integer
    cx As Integer
    cy As Integer
End Type
```

```
Type DOC_INFO_1
    pDocName As String
    pOutputFile As String
    pDatatype As String
End Type
```

```
Type DOC_INFO_2
    pDocName As String
    pOutputFile As String
    pDatatype As String
    dwMode As Long
    JobId As Long
End Type
```

```
Type DOCINFO
    cbSize As Long
    lpszDocName As String
    lpszOutput As String
End Type
```

```
Type DRAGINFO
    uSize As Long
    pt As POINTAPI
    fNC As Boolean
    lpFileList As String
    grfKeyState As Long
End Type
```

```
Type DRAWITEMSTRUCT
    CtlType As Long
    CtlID As Long
    itemID As Long
    itemAction As Long
    itemState As Long
    hwndItem As Long
    hdc As Long
    rcItem As Rect
    itemData As Long
End Type
```

```
Type DRAWTEXTPARAMS
    cbSize As Long
    iTabLength As Long
```

```
    iLeftMargin As Long
    iRightMargin As Long
    uiLengthDrawn As Long
End Type
```

Type **DRIVER_INFO_1**
```
    pName As String
End Type
```

Type **DRIVER_INFO_2**
```
    cVersion As Long
    pName As String
    pEnvironment As String
    pDriverPath As String
    pDataFile As String
    pConfigFile As String
End Type
```

Type **DRIVER_INFO_3**
```
    cVersion As Long
    pName As String
    pEnvironment As String
    pDriverPath As String
    pDataFile As String
    pConfigFile As String
    pHelpFile As String
    pDependentFiles As String
    pMonitorName As String
    pDefaultDataType As String
End Type
```

Type **DRVCONFIGINFO**
```
    dwDCISize As Long
    lpszDCISectionName As String
    lpszDCIAliasName As String
    dnDevNode As Long
End Type
```

Type **emr**
```
    iType As Long
    nSize As Long
End Type
```

Type **EMRABORTPATH**
```
    pEmr As emr
End Type
```

Type **EMRANGLEARC**
```
    pEmr As emr
    ptlCenter As POINTL
    nRadius As Long
    eStartAngle As Double
    eSweepAngle As Double
End Type
```

Type **EMRARC**
```
    pEmr As emr
    rclBox As RECTL
    ptlStart As POINTL
    ptlEnd As POINTL
End Type
```

Type **EMRARCTO**
```
    pEmr As emr
    rclBox As RECTL
    ptlStart As POINTL
    ptlEnd As POINTL
End Type
```

Type **EMRBEGINPATH**
```
    pEmr As emr
End Type
```

Type **EMRBITBLT**
```
    pEmr As emr
    rclBounds As RECTL
    xDest As Long
    yDest As Long
    cxDest As Long
    cyDest As Long
    dwRop As Long
    xSrc As Long
    ySrc As Long
    xformSrc As xform
    crBkColorSrc As Long
    iUsageSrc As Long
    offBmiSrc As Long
    cbBmiSrc As Long
    offBitsSrc As Long
    cbBitsSrc As Long
End Type
```

Type **EMRCHORD**
```
    pEmr As emr
    rclBox As RECTL
    ptlStart As POINTL
    ptlEnd As POINTL
End Type
```

Type **EMRCLOSEFIGURE**
```
    pEmr As emr
End Type
```

Type **EMRCREATEBRUSHIN DIRECT**
```
    pEmr As emr
    ihBrush As Long
    lb As LOGBRUSH
End Type
```

Type **EMRCREATECOLORSPACE**
```
    pEmr As emr
    ihCS As Long
    lcs As LOGCOLORSPACE
End Type
```

Type **EMRCREATEDIB PATTERNBRUSHPT**
```
    pEmr As emr
    ihBursh As Long
    iUsage As Long
    offBmi As Long
    cbBmi As Long
    offBits As Long
    cbBits As Long
End Type
```

Type **EMRCREATEMONOBRUSH**
```
    pEmr As emr
    ihBrush As Long
    iUsage As Long
    offBmi As Long
    cbBmi As Long
    offBits As Long
    cbBits As Long
End Type
```

Type **EMRCREATEPALETTE**
```
    pEmr As emr
    ihPal As Long
    lgpl As LOGPALETTE
End Type
```

Type **EMRCREATEPEN**
```
    pEmr As emr
    ihPen As Long
    lopn As LOGPEN
End Type
```

Type **EMRDELETEOBJECT**
```
    pEmr As emr
    ihObject As Long
End Type
```

Type **EMRELLIPSE**
```
    pEmr As emr
    rclBox As RECTL
End Type
```

Type **EMREMRSAVEDC**
```
    pEmr As emr
End Type
```

```
Type EMRENDPATH
    pEmr As emr
End Type
```

```
Type EMREOF
    pEmr As emr
    nPalEntries As Long
    offPalEntries As Long
    nSizeLast As Long
End Type
```

```
Type EMREXCLUDECLIPRECT
    pEmr As emr
    rclClip As RECTL
End Type
```

```
Type EMREXTCREATE
                    FONTINDIRECT
    pEmr As emr
    ihFont As Long
    elfw As EXTLOGFONT
End Type
```

```
Type EMREXTCREATEPEN
    pEmr As emr
    ihPen As Long
    offBmi As Long
    cbBmi As Long
    offBits As Long
    cbBits As Long
    elp As EXTLOGPEN
End Type
```

```
Type EMREXTFLOODFILL
    pEmr As emr
    ptlStart As POINTL
    crColor As Long
    iMode As Long
End Type
```

```
Type EMREXTSELECTCLIPRGN
    pEmr As emr
    cbRgnData As Long
    iMode As Long
    RgnData(1) As Integer
End Type
```

```
Type EMREXTTEXTOUT
    pEmr As emr
    rclBounds As RECTL
    iGraphicsMode As Long
    exScale As Double
    eyScale As Double
    emrtext As emrtext
End Type
```

```
Type EMRFILLPATH
    pEmr As emr
    rclBounds As RECTL
End Type
```

```
Type EMRFILLRGN
    pEmr As emr
    rclBounds As RECTL
    cbRgnData As Long
    ihBrush As Long
    RgnData(1) As Integer
End Type
```

```
Type EMRFLATTENPATH
    pEmr As emr
End Type
```

```
Type EMRFRAMERGN
    pEmr As emr
    rclBounds As RECTL
    cbRgnData As Long
    ihBrush As Long
    szlStroke As SIZEL
    RgnData(1) As Integer
End Type
```

```
Type EMRGDICOMMENT
    pEmr As emr
    cbData As Long
    Data(1) As Integer
End Type
```

```
Type EMRINTERSECTCLIPRECT
    pEmr As emr
    rclClip As RECTL
End Type
```

```
Type EMRINVERTRGN
    pEmr As emr
    rclBounds As RECTL
    cbRgnData As Long
    RgnData(1) As Integer
End Type
```

```
Type EMRLINETO
    pEmr As emr
    ptl As POINTL
End Type
```

```
Type EMRMASKBLT
    pEmr As emr
    rclBounds As RECTL
    xDest As Long
    yDest As Long
    cxDest As Long
    cyDest As Long
    dwRop As Long
    xSrc2 As Long
    cyDest2 As Long
    dwRop2 As Long
    xSrc As Long
    ySrc As Long
    xformSrc As XFORM
    crBkColorSrc As Long
    iUsageSrc As Long
    offBmiSrc As Long
    cbBmiSrc As Long
    offBitsSrc As Long
    cbBitsSrc As Long
    xMask As Long
    yMask As Long
    iUsageMask As Long
    offBmiMask As Long
    cbBmiMask As Long
    offBitsMask As Long
    cbBitsMask As Long
End Type
```

```
Type EMRMODIFYWORLDTRANSFORM
    pEmr As emr
    xform As XFORM
    iMode As Long
End Type
```

```
Type EMRMOVETOEX
    pEmr As emr
    ptl As POINTL
End Type
```

```
Type EMROFFSETCLIPRGN
    pEmr As emr
    ptlOffset As POINTL
End Type
```

```
Type EMRPAINTRGN
    pEmr As emr
    rclBounds As RECTL
    cbRgnData As Long
    RgnData(1) As Integer
End Type
```

```
Type EMRPIE
    pEmr As emr
    rclBox As RECTL
    ptlStart As POINTL
    ptlEnd As POINTL
End Type
```

```
Type EMRPLGBLT
    pEmr As emr
    rclBounds As RECTL
    aptlDest(3) As POINTL
    xSrc As Long
    ySrc As Long
    cxSrc As Long
    cySrc As Long
    xformSrc As XFORM
    crBkColorSrc As Long
    iUsageSrc As Long
    offBmiSrc As Long
    cbBmiSrc As Long
    offBitsSrc As Long
    cbBitsSrc As Long
    xMask As Long
    yMask As Long
    iUsageMask As Long
    offBmiMask As Long
    cbBmiMask As Long
    offBitsMask As Long
    cbBitsMask As Long
End Type
```

```
Type EMRPLOYBEZIERTO16
    pEmr As emr
    rclBounds As RECTL
    cpts As Long
    apts(1) As POINTS
End Type
```

```
Type EMRPOLYBEZIER
    pEmr As emr
    rclBounds As RECTL
    cptl As Long
    aptl(1) As POINTL
End Type
```

```
Type EMRPOLYBEZIER16
    pEmr As emr
    rclBounds As RECTL
    cpts As Long
    apts(1) As POINTS
End Type
```

```
Type EMRPOLYBEZIERTO
    pEmr As emr
    rclBounds As RECTL
    cptl As Long
    aptl(1) As POINTL
End Type
```

```
Type EMRPOLYDRAW
    pEmr As emr
    rclBounds As RECTL
    cptl As Long
    aptl(1) As POINTL
    abTypes(1) As Integer
End Type
```

```
Type EMRPOLYDRAW16
    pEmr As emr
    rclBounds As RECTL
    cpts As Long
    apts(1) As POINTS
    abTypes(1) As Integer
End Type
```

```
Type EMRPOLYGON
    pEmr As emr
    rclBounds As RECTL
    cptl As Long
    aptl(1) As POINTL
End Type
```

```
Type EMRPOLYGON16
    pEmr As emr
    rclBounds As RECTL
    cpts As Long
    apts(1) As POINTS
End Type
```

```
Type EMRPOLYLINE
    pEmr As emr
    rclBounds As RECTL
    cptl As Long
    aptl(1) As POINTL
End Type
```

```
Type EMRPOLYLINE16
    pEmr As emr
    rclBounds As RECTL
    cpts As Long
    apts(1) As POINTS
End Type
```

```
Type EMRPOLYLINETO16
    pEmr As emr
    rclBounds As RECTL
    cpts As Long
    apts(1) As POINTS
End Type
```

```
Type EMRPOLYPOLYGON
    pEmr As emr
    rclBounds As RECTL
    nPolys As Long
    cptl As Long
    aPolyCounts(1) As Long
    aptl(1) As POINTL
End Type
```

```
Type EMRPOLYPOLYGON16
    pEmr As emr
    rclBounds As RECTL
    nPolys As Long
    cpts As Long
    aPolyCounts(1) As Long
    apts(1) As POINTS
End Type
```

```
Type EMRPOLYPOLYLINE
    pEmr As emr
    rclBounds As RECTL
    nPolys As Long
    cptl As Long
    aPolyCounts(1) As Long
    aptl(1) As POINTL
End Type
```

```
Type EMRPOLYPOLYLINE16
    pEmr As emr
    rclBounds As RECTL
    nPolys As Long
    cpts As Long
    aPolyCounts(1) As Long
    apts(1) As POINTS
End Type
```

```
Type EMRREALIZEPALETTE
    pEmr As emr
End Type
```

```
Type EMRRECTANGLE
    pEmr As emr
    rclBox As RECTL
End Type
```

```
Type EMRRESIZEPALETTE
    pEmr As emr
    ihPal As Long
    cEntries As Long
End Type
```

```
Type EMRRESTOREDC
    pEmr As emr
    iRelative As Long
End Type
```

```
Type EMRROUNDRECT
    pEmr As emr
    rclBox As RECTL
    szlCorner As SIZEL
End Type
```

```
Type
    EMRSCALEVIEWPORTEXTEX
    pEmr As emr
    xNum As Long
    xDenom As Long
    yNum As Long
    yDemon As Long
End Type
```

```
Type EMRSCALEWINDOWEXTEX
    pEmr As emr
    xNum As Long
    xDenom As Long
    yNum As Long
    yDemon As Long
End Type
```

```
Type EMRSELECTCLIPPATH
    pEmr As emr
    iMode As Long
End Type
```

```
Type EMRSELECTCOLORSPACE
    pEmr As emr
    ihCS As Long
End Type
```

```
Type EMRSELECTOBJECT
    pEmr As emr
    ihObject As Long
End Type
```

```
Type EMRSELECTPALETTE
    pEmr As emr
    ihPal As Long
End Type
```

```
Type EMRSETARCDIRECTION
    pEmr As emr
    iArcDirection As Long
End Type
```

```
Type EMRSETBKCOLOR
    pEmr As emr
    crColor As Long
End Type
```

```
Type EMRSETBKMODE
    pEmr As emr
    iMode As Long
End Type
```

```
Type EMRSETBRUSHORGEX
    pEmr As emr
    ptlOrigin As POINTL
End Type
```

```
Type
    EMRSETCOLORADJUSTMENT
    pEmr As emr
    ColorAdjustment As
ColorAdjustment
End Type
```

```
Type EMRSETDIBITSTODEVICE
    pEmr As emr
    rclBounds As RECTL
    xDest As Long
    yDest As Long
    xSrc As Long
    ySrc As Long
    cxSrc As Long
    cySrc As Long
    offBmiSrc As Long
    cbBmiSrc As Long
    offBitsSrc As Long
    cbBitsSrc As Long
    iUsageSrc As Long
    iStartScan As Long
    cScans As Long
End Type
```

```
Type EMRSETMAPMODE
    pEmr As emr
    iMode As Long
End Type
```

```
Type EMRSETMAPPERFLAGS
    pEmr As emr
    dwFlags As Long
End Type
```

```
Type EMRSETMETARGN
    pEmr As emr
End Type
```

```
Type EMRSETMITERLIMIT
    pEmr As emr
    eMiterLimit As Double
End Type
```

```
Type EMRSETPALETTEENTRIES
    pEmr As emr
    ihPal As Long
    iStart As Long
    cEntries As Long
    aPalEntries(1) As PALETTEENTRY
End Type
```

```
Type EMRSETPIXELV
    pEmr As emr
    ptlPixel As POINTL
    crColor As Long
End Type
```

```
Type EMRSETPOLYFILLMODE
    pEmr As emr
    iMode As Long
End Type
```

```
Type EMRSETROP2
    pEmr As emr
    iMode As Long
End Type
```

```
Type EMRSETSTRETCHBLTMODE
    pEmr As emr
    iMode As Long
End Type
```

```
Type EMRSETTEXTALIGN
    pEmr As emr
    iMode As Long
End Type
```

```
Type EMRSETTEXTCOLOR
    pEmr As emr
    crColor As Long
End Type
```

```
Type EMRSETVIEWPORTEXTEX
    pEmr As emr
    szlExtent As SIZEL
End Type
```

```
Type EMRSETVIEWPORTORGEX
    pEmr As emr
    ptlOrigin As POINTL
End Type
```

```
Type EMRSETWINDOWEXTEX
    pEmr As emr
    szlExtent As SIZEL
End Type
```

```
Type EMRSETWINDOWORGEX
    pEmr As emr
    ptlOrigin As POINTL
End Type
```

```
Type EMRSETWORLDTRANSFORM
    pEmr As emr
    xform As XFORM
End Type
```

```
Type EMRSTRETCHBLT
    pEmr As emr
    rclBounds As RECTL
    xDest As Long
    yDest As Long
    cxDest As Long
    cyDest As Long
    dwRop As Long
    xSrc As Long
    ySrc As Long
    xformSrc As xform
    crBkColorSrc As Long
    iUsageSrc As Long
    offBmiSrc As Long
    cbBmiSrc As Long
    offBitsSrc As Long
    cbBitsSrc As Long
    cxSrc As Long
    cySrc As Long
End Type
```

```
Type EMRSTRETCHDIBITS
    pEmr As emr
    rclBounds As RECTL
    xDest As Long
    yDest As Long
    xSrc As Long
    ySrc As Long
    cxSrc As Long
    cySrc As Long
    offBmiSrc As Long
    cbBmiSrc As Long
    offBitsSrc As Long
    cbBitsSrc As Long
    iUsageSrc As Long
    dwRop As Long
    cxDest As Long
    cyDest As Long
End Type
```

```
Type EMRSTROKEANDFILLPATH
    pEmr As emr
    rclBounds As RECTL
End Type
```

```
Type EMRSTROKEPATH
    pEmr As emr
    rclBounds As RECTL
End Type
```

```
Type emrtext
    ptlReference As POINTL
    nchars As Long
    offString As Long
    fOptions As Long
    rcl As RECTL
    offDx As Long
End Type
```

```
Type EMRWIDENPATH
    pEmr As emr
End Type
```

```
Type ENHMETAHEADER
    iType As Long
    nSize As Long
    rclBounds As RECTL
    rclFrame As RECTL
    dSignature As Long
    nVersion As Long
    nBytes As Long
    nRecords As Long
    nHandles As Integer
    sReserved As Integer
    nDescription As Long
    offDescription As Long
    nPalEntries As Long
    szlDevice As SIZEL
    szlMillimeters As SIZEL
End Type
```

```
Type ENHMETARECORD
    iType As Long
    nSize As Long
    dParm(1) As Long
End Type
```

```
Type ENUM_SERVICE_STATUS
    lpServiceName As String
    lpDisplayName As String
    ServiceStatus As
                    SERVICE_STATUS
End Type
```

```
Type ENUMLOGFONT
    elfLogFont As LOGFONT
    elfFullName(LF_FULLFACE
                    SIZE) As Byte
    elfStyle(LF_FACESIZE) As Byte
End Type
```

```
Type ENUMLOGFONTEX
    elfLogFont As LOGFONT
    elfFullName(LF_FULLFACE
                    SIZE) As Byte
    elfStyle(LF_FACESIZE) As Byte
    elfScript(LF_FACESIZE) As Byte
End Type
```

```
Type EVENTLOGRECORD
    LengthasLong
    ReservedasLong
    RecordNumberasLong
    TimeGeneratedasLong
    TimeWrittenasLong
    EventIDasLong
    EventTypeasInteger
    NumStringsasInteger
    EventCategoryasInteger
    ReservedFlagsasInteger
    ClosingRecordNumberasLong
    StringOffsetasLong
    UserSidLengthasLong
    UserSidOffsetasLong
    DataLengthasLong
    DataOffsetasLong
End Type
```

```
Type EVENTMSG
    message As Long
    paramL As Long
    paramH As Long
    time As Long
    hwnd As Long
End Type
```

```
Type EXCEPTION_DEBUG_INFO
    pExceptionRecord As
                EXCEPTION_RECORD
    dwFirstChance As Long
End Type
```

```
Type EXCEPTION_POINTERS
    pExceptionRecord As
                EXCEPTION_RECORD
    ContextRecord As CONTEXT
End Type
```

```
Type EXCEPTION_RECORD
    ExceptionCode As Long
    ExceptionFlags As Long
    pExceptionRecord As Long
    ExceptionAddress As Long
    NumberParameters As Long
```

VisualBasic用構造体定義一覧

```
    ExceptionInformation(
        EXCEPTION_MAXIMUM_
            PARAMETERS) As Long
End Type
```

```
Type EXIT_PROCESS_DEBUG_INFO
    dwExitCode As Long
End Type
```

```
Type EXIT_THREAD_DEBUG_INFO
    dwExitCode As Long
End Type
```

```
Type EXTLOGFONT
    elfLogFont As LOGFONT
    elfFullName(LF_FULLFACE
                    SIZE) As Byte
    elfStyle(LF_FACESIZE) As Byte
    elfVersion As Long
    elfStyleSize As Long
    elfMatch As Long
    elfReserved As Long
    elfVendorId(ELF_VENDOR_
                    SIZE) As Byte
    elfCulture As Long
    elfPanose As PANOSE
End Type
```

```
Type EXTLOGPEN
    elpPenStyle As Long
    elpWidth As Long
    elpBrushStyle As Long
    elpColor As Long
    elpHatch As Long
    elpNumEntries As Long
    elpStyleEntry(1) As Long
End Type
```

```
Type FILETIME
    dwLowDateTime As Long
    dwHighDateTime As Long
End Type
```

```
Type FIND_NAME_BUFFER
    Length As Integer
    access_control As Integer
    frame_control As Integer
    destination_addr(6) As Integer
    source_addr(6) As Integer
    routing_info(18) As Integer
End Type
```

```
Type FIND_NAME_HEADER
    node_count As Integer
    Reserved As Integer
    unique_group As Integer
End Type
```

```
Type FINDREPLACE
    lStructSize As Long
    hwndOwner As Long
    hInstance As Long
    flags As Long
    lpstrFindWhat As String
    lpstrReplaceWith As String
    wFindWhatLen As Integer
    wReplaceWithLen As Integer
    lCustData As Long
    lpfnHook As Long
    lpTemplateName As String
End Type
```

```
Type FIXED
    fract As Integer
    Value As Integer
End Type
```

```
Type FOCUS_EVENT_RECORD
    bSetFocus As Boolean
End Type
```

```
Type FONTSIGNATURE
    fsUsb(4) As Long
    fsCsb(2) As Long
End Type
```

```
Type FORM_INFO_1
    pName As String
    Size As SIZEL
    ImageableArea As RECTL
End Type
```

```
Type GCP_RESULTS
    lStructSize As Long
    lpOutString As String
    lpOrder As Long
    lpDX As Long
    lpCaretPos As Long
    lpClass As String
    lpGlyphs As String
    nGlyphs As Long
    nMaxFit As Long
End Type
```

```
Type GENERIC_MAPPING
    GenericRead As Long
    GenericWrite As Long
    GenericExecute As Long
    GenericAll As Long
End Type
```

```
Type GLYPHMETRICS
    gmBlackBoxX As Long
    gmBlackBoxY As Long
    gmptGlyphOrigin As POINTAPI
    gmCellIncX As Integer
    gmCellIncY As Integer
End Type
```

```
Type HANDLETABLE
    objectHandle(1) As Long
End Type
```

```
Type HELPINFO
    cbSize As Long
    iContextType As Long
    iCtrlId As Long
    hItemHandle As Long
    dwContextId As Long
    MousePos As POINTAPI
End Type
```

```
Type HELPWININFO
    wStructSize As Long
    x As Long
    y As Long
    dx As Long
    dy As Long
    wMax As Long
    rgchMember As String*2
End Type
```

```
Type HSZPAIR
    hszSvc As Long
    hszTopic As Long
End Type
```

```
Type ICONINFO
    fIcon As Boolean
    xHotspot As Long
    yHotspot As Long
    hbmMask As Long
    hbmColor As Long
End Type
```

```
Type ICONMETRICS
    cbSize As Long
    iHorzSpacing As Long
    iVertSpacing As Long
    iTitleWrap As Long
    lfFont As LOGFONT
End Type
```

```
Type JOB_INFO_1
    JobId As Long
    pPrinterName As String
    pMachineName As String
    pUserName As String
    pDocument As String
    pDatatype As String
    pStatus As String
    Status As Long
    Priority As Long
    Position As Long
    TotalPages As Long
    PagesPrinted As Long
    Submitted As SYSTEMTIME
End Type
```

```
Type JOB_INFO_2
    JobId As Long
    pPrinterName As String
    pMachineName As String
    pUserName As String
    pDocument As String
    pNotifyName As String
    pDatatype As String
    pPrintProcessor As String
    pParameters As String
    pDriverName As String
    pDevMode As DEVMODE
    pStatus As String
    pSecurityDescriptor As SECU
                    RITY_DESCRIPTOR
    Status As Long
    Priority As Long
    Position As Long
    StartTime As Long
    UntilTime As Long
    TotalPages As Long
    Size As Long
    Submitted As SYSTEMTIME
    time As Long
    PagesPrinted As Long
End Type
```

```
Type JOYCAPS
    wMid As Integer
    wPid As Integer
    szPname As
            String*MAXPNAMELEN
    wXmin As Integer
    wXmax As Integer
    wYmin As Integer
    wYmax As Integer
    wZmin As Integer
    wZmax As Integer
    wNumButtons As Integer
    wPeriodMin As Integer
    wPeriodMax As Integer
End Type
```

```
Type JOYINFO
    wXpos As Integer
    wYpos As Integer
    wZpos As Integer
    wButtons As Integer
End Type
```

```
Type JOYINFOEX
    dwSize As Long
    dwFlags As Long
    dwXpos As Long
    dwYpos As Long
    dwZpos As Long
    dwRpos As Long
    dwUpos As Long
    dwVpos As Long
    dwButtons As Long
    dwButtonNumber As Long
    dwPOV As Long
    dwReserved1 As Long
    dwReserved2 As Long
End Type
```

```
Type KERNINGPAIR
    wFirst As Integer
    wSecond As Integer
    iKernAmount As Long
End Type
```

```
Type KEY_EVENT_RECORD
    bKeyDown As Boolean
    wRepeatCount As Integer
    wVirtualKeyCode As Integer
    wVirtualScanCode As Integer
    uChar As Integer
    dwControlKeyState As Long
End Type
```

```
Type LANA_ENUM
    Length As Integer
    lana(MAX_LANA) As Integer
End Type
```

```
Type LARGE_INTEGER
    lowpart As Long
    highpart As Long
End Type
```

```
Type LDT_BYTES
    BaseMid As Byte
    Flags1 As Byte
    Flags2 As Byte
    BaseHi As Byte
End Type
```

```
Type LDT_ENTRY
    LimitLow As Integer
    BaseLow As Integer
    HighWord As Long
End Type
```

```
Type LOAD_DLL_DEBUG_INFO
    hFile As Long
    lpBaseOfDll As Long
    dwDebugInfoFileOffset As Long
    nDebugInfoSize As Long
    lpImageName As Long
    fUnicode As Integer
End Type
```

```
Type LOCALESIGNATURE
    lsUsb(4) As Long
    lsCsbDefault(2) As Long
    lsCsbSupported(2) As Long
End Type
```

```
Type LOGBRUSH
    lbStyle As Long
    lbColor As Long
    lbHatch As Long
End Type
```

```
Type LOGCOLORSPACE
    lcsSignature As Long
    lcsVersion As Long
    lcsSize As Long
    lcsCSType As Long
    lcsIntent As Long
    lcsEndPoints As CIEXYZTRIPLE
    lcsGammaRed As Long
    lcsGammaGreen As Long
    lcsGammaBlue As Long
    lcsFileName As
                String*MAX_PATH
End Type
```

```
Type LOGFONT
    lfHeight As Long
    lfWidth As Long
    lfEscapement As Long
    lfOrientation As Long
```

```
    lfWeight As Long
    lfItalic As Byte
    lfUnderline As Byte
    lfStrikeOut As Byte
    lfCharSet As Byte
    lfOutPrecision As Byte
    lfClipPrecision As Byte
    lfQuality As Byte
    lfPitchAndFamily As Byte
    lfFaceName(LF_FACESIZE)
                        As Byte
End Type
```

Type **LOGPALETTE**
```
    palVersion As Integer
    palNumEntries As Integer
    palPalEntry(1)
              As PALETTEENTRY
End Type
```

Type **LOGPEN**
```
    lopnStyle As Long
    lopnWidth As POINTAPI
    lopnColor As Long
End Type
```

Type **LUID**
```
    LowPart As Long
    HighPart As Long
End Type
```

Type **LUID_AND_ATTRIBUTES**
```
    pLuid As LUID
    Attributes As Long
End Type
```

Type **MAT2**
```
    eM11 As FIXED
    eM12 As FIXED
    eM21 As FIXED
    eM22 As FIXED
End Type
```

Type **MCI_ANIM_OPEN_PARMS**
```
    dwCallback As Long
    wDeviceID As Long
    lpstrDeviceType As String
    lpstrElementName As String
    lpstrAlias As String
    dwStyle As Long
    hWndParent As Long
End Type
```

Type **MCI_ANIM_PLAY_PARMS**
```
    dwCallback As Long
    dwFrom As Long
    dwTo As Long
    dwSpeed As Long
End Type
```

Type **MCI_ANIM_RECT_PARMS**
```
    dwCallback As Long
    rc As Rect
End Type
```

Type **MCI_ANIM_STEP_PARMS**
```
    dwCallback As Long
    dwFrames As Long
End Type
```

Type **MCI_ANIM_UPDATE_PARMS**
```
    dwCallback As Long
    rc As Rect
    hdc As Long
End Type
```

Type **MCI_ANIM_WINDOW_PARMS**
```
    dwCallback As Long
    hwnd As Long
    nCmdShow As Long
    lpstrText As String
End Type
```

Type **MCI_BREAK_PARMS**
```
    dwCallback As Long
    nVirtKey As Long
    hwndBreak As Long
End Type
```

Type **MCI_GENERIC_PARMS**
```
    dwCallback As Long
End Type
```

Type **MCI_GETDEVCAPS_PARMS**
```
    dwCallback As Long
    dwReturn As Long
    dwIten As Long
End Type
```

Type **MCI_INFO_PARMS**
```
    dwCallback As Long
    lpstrReturn As String
    dwRetSize As Long
End Type
```

Type **MCI_LOAD_PARMS**
```
    dwCallback As Long
    lpFileName As String
End Type
```

Type **MCI_OPEN_PARMS**
```
    dwCallback As Long
    wDeviceID As Long
    lpstrDeviceType As String
    lpstrElementName As String
    lpstrAlias As String
End Type
```

Type **MCI_OVLY_LOAD_PARMS**
```
    dwCallback As Long
    lpFileName As String
    rc As Rect
End Type
```

Type **MCI_OVLY_OPEN_PARMS**
```
    dwCallback As Long
    wDeviceID As Long
    lpstrDeviceType As String
    lpstrElementName As String
    lpstrAlias As String
    dwStyle As Long
    hWndParent As Long
End Type
```

Type **MCI_OVLY_RECT_PARMS**
```
    dwCallback As Long
    rc As Rect
End Type
```

Type **MCI_OVLY_SAVE_PARMS**
```
    dwCallback As Long
    lpFileName As String
    rc As Rect
End Type
```

Type **MCI_OVLY_WINDOW_PARMS**
```
    dwCallback As Long
    hwnd As Long
    nCmdShow As Long
    lpstrText As String
End Type
```

Type **MCI_PLAY_PARMS**
```
    dwCallback As Long
    dwFrom As Long
    dwTo As Long
End Type
```

Type **MCI_RECORD_PARMS**
```
    dwCallback As Long
    dwFrom As Long
    dwTo As Long
End Type
```

```
Type MCI_SAVE_PARMS
    dwCallback As Long
    lpFileName As String
End Type

Type MCI_SEEK_PARMS
    dwCallback As Long
    dwTo As Long
End Type

Type MCI_SEQ_SET_PARMS
    dwCallback As Long
    dwTimeFormat As Long
    dwAudio As Long
    dwTempo As Long
    dwPort As Long
    dwSlave As Long
    dwMaster As Long
    dwOffset As Long
End Type

Type MCI_SET_PARMS
    dwCallback As Long
    dwTimeFormat As Long
    dwAudio As Long
End Type

Type MCI_SOUND_PARMS
    dwCallback As Long
    lpstrSoundName As String
End Type

Type MCI_STATUS_PARMS
    dwCallback As Long
    dwReturn As Long
    dwItem As Long
    dwTrack As Integer
End Type

Type MCI_SYSINFO_PARMS
    dwCallback As Long
    lpstrReturn As String
    dwRetSize As Long
    dwNumber As Long
    wDeviceType As Long
End Type

Type MCI_VD_ESCAPE_PARMS
    dwCallback As Long
    lpstrCommand As String
End Type

Type MCI_VD_PLAY_PARMS
    dwCallback As Long
    dwFrom As Long
    dwTo As Long
    dwSpeed As Long
End Type

Type MCI_VD_STEP_PARMS
    dwCallback As Long
    dwFrames As Long
End Type

Type MCI_WAVE_DELETE_PARMS
    dwCallback As Long
    dwFrom As Long
    dwTo As Long
End Type

Type MCI_WAVE_OPEN_PARMS
    dwCallback As Long
    wDeviceID As Long
    lpstrDeviceType As String
    lpstrElementName As String
    lpstrAlias As String
    dwBufferSeconds As Long
End Type

Type MCI_WAVE_SET_PARMS
    dwCallback As Long
    dwTimeFormat As Long
    dwAudio As Long
    wInput As Long
    wOutput As Long
    wFormatTag As Integer
    wReserved2 As Integer
    nChannels As Integer
    wReserved3 As Integer
    nSamplesPerSec As Long
    nAvgBytesPerSec As Long
    nBlockAlign As Integer
    wReserved4 As Integer
    wBitsPerSample As Integer
    wReserved5 As Integer
End Type

Type MDICREATESTRUCT
    szClass As String
    szTitle As String
    hOwner As Long
    x As Long
    y As Long
    cx As Long
    cy As Long
    style As Long
    lParam As Long
End Type

Type MEASUREITEMSTRUCT
    CtlType As Long
    CtlID As Long
    itemID As Long
    itemWidth As Long
    itemHeight As Long
    itemData As Long
End Type

Type MEMORY_BASIC_INFORMATION
    BaseAddressasLong
    AllocationBaseasLong
    AllocationProtect As Long
    RegionSize As Long
    State As Long
    Protect As Long
    lType As Long
End Type

Type MEMORYSTATUS
    dwLength As Long
    dwMemoryLoad As Long
    dwTotalPhys As Long
    dwAvailPhys As Long
    dwTotalPageFile As Long
    dwAvailPageFile As Long
    dwTotalVirtual As Long
    dwAvailVirtual As Long
End Type

Type MENU_EVENT_RECORD
    dwCommandId As Long
End Type

Type MENUITEMINFO
    cbSize As Long
    fMask As Long
    fType As Long
    fState As Long
    wID As Long
    hSubMenu As Long
    hbmpChecked As Long
    hbmpUnchecked As Long
    dwItemData As Long
    dwTypeData As String
    cch As Long
End Type

Type MENUITEMTEMPLATE
    mtOption As Integer
    mtID As Integer
    mtString As Byte
End Type
```

```
Type MENUITEMTEMPLATEHEADER
    versionNumber As Integer
    offset As Integer
End Type
```

```
Type METAFILEPICT
    mm As Long
    xExt As Long
    yExt As Long
    hMF As Long
End Type
```

```
Type METAHEADER
    mtType As Integer
    mtHeaderSize As Integer
    mtVersion As Integer
    mtSize As Long
    mtNoObjects As Integer
    mtMaxRecord As Long
    mtNoParameters As Integer
End Type
```

```
Type METARECORD
    rdSize As Long
    rdFunction As Integer
    rdParm(1) As Integer
End Type
```

```
Type midi
    songptrpos As Long
End Type
```

```
Type MIDIEVENT
    dwDeltaTime As Long
    dwStreamID As Long
    dwEvent As Long
    dwParms(1) As Long
End Type
```

```
Type MIDIHDR
    lpData As String
    dwBufferLength As Long
    dwBytesRecorded As Long
    dwUser As Long
    dwFlags As Long
    lpNext As Long
    Reserved As Long
End Type
```

```
Type MIDIINCAPS
    wMid As Integer
    wPid As Integer
    vDriverVersion As Long
    szPname As
                String*MAXPNAMELEN
End Type
```

```
Type MIDIOUTCAPS
    wMid As Integer
    wPid As Integer
    vDriverVersion As Long
    szPname As
                String*MAXPNAMELEN
    wTechnology As Integer
    wVoices As Integer
    wNotes As Integer
    wChannelMask As Integer
    dwSupport As Long
End Type
```

```
Type MIDIPROPTEMPO
    cbStruct As Long
    dwTempo As Long
End Type
```

```
Type MIDIPROPTIMEDIV
    cbStruct As Long
    dwTimeDiv As Long
End Type
```

```
Type MIDISTRMBUFFVER
    dwVersion As Long
    dwMid As Long
    dwOEMVersion As Long
End Type
```

```
Type MINIMIZEDMETRICS
    cbSize As Long
    iWidth As Long
    iHorzGap As Long
    iVertGap As Long
    iArrange As Long
    lfFont As LOGFONT
End Type
```

```
Type MINMAXINFO
    ptReserved As POINTAPI
    ptMaxSize As POINTAPI
    ptMaxPosition As POINTAPI
    ptMinTrackSize As POINTAPI
    ptMaxTrackSize As POINTAPI
End Type
```

```
Type MIXERCAPS
    wMid As Integer
    wPid As Integer
    vDriverVersion As Long
    szPname As String*MAXPNAMELEN
    fdwSupport As Long
    cDestinations As Long
End Type
```

```
Type MIXERCONTROL
    cbStruct As Long
    dwControlID As Long
    dwControlType As Long
    fdwControl As Long
    cMultipleItems As Long
    szShortName As String*MIXER_
                SHORT_NAME_CHARS
    szName As String*MIXER_
                LONG_NAME_CHARS
    Bounds As Double
    Metrics As Long
End Type
```

```
Type MIXERCONTROLDETAILS
    cbStruct As Long
    dwControlID As Long
    cChannels As Long
    item As Long
    cbDetails As Long
    paDetails As Long
End Type
```

```
Type MIXERCONTROL
                DETAILS_BOOLEAN
    fValue As Long
End Type
```

```
Type MIXERCONTROL
                DETAILS_LISTTEXT
    dwParam1 As Long
    dwParam2 As Long
    szName As
String*MIXER_LONG_NAME_CHA
RS
End Type
```

```
Type MIXERCONTROL
                DETAILS_SIGNED
    lValue As Long
End Type
```

```
Type MIXERCONTROL
                DETAILS_UNSIGNED
    dwValue As Long
End Type
```

```
Type MIXERLINE
    cbStruct As Long
    dwDestination As Long
    dwSource As Long
    dwLineID As Long
    fdwLine As Long
```

```
    dwUser As Long
    dwComponentType As Long
    cChannels As Long
    cConnections As Long
    cControls As Long
    szShortName As String*MIXER
              _SHORT_NAME_CHARS
    szName As String*MIXER
              _LONG_NAME_CHARS
    lpTarget As Target
End Type
```

Type **MIXERLINECONTROLS**
```
    cbStruct As Long
    dwLineID As Long
    dwControl As Long
    cControls As Long
    cbmxctrl As Long
    pamxctrl As MIXERCONTROL
End Type
```

Type **MMCKINFO**
```
    ckid As Long
    ckSize As Long
    fccType As Long
    dwDataOffset As Long
    dwFlags As Long
End Type
```

Type **MMIOINFO**
```
    dwFlags As Long
    fccIOProc As Long
    pIOProc As Long
    wErrorRet As Long
    htask As Long
    cchBuffer As Long
    pchBuffer As String
    pchNext As String
    pchEndRead As String
    pchEndWrite As String
    lBufOffset As Long
    lDiskOffset As Long
    adwInfo(4) As Long
    dwReserved1 As Long
    dwReserved2 As Long
    hmmio As Long
End Type
```

Type **MMTIME**
```
    wType As Long
    u As Long
End Type
```

Type **MODEMDEVCAPS**
```
    dwActualSize As Long
    dwRequiredSize As Long
    dwDevSpecificOffset As Long
    dwDevSpecificSize As Long
    dwModemProviderVersion As Long
    dwModemManufacturerOffse
                          t As Long
    dwModemManufacturerSize
                           As Long
    dwModemModelOffset As Long
    dwModemModelSize As Long
    dwModemVersionOffset As Long
    dwModemVersionSize As Long
    dwDialOptions As Long
    dwCallSetupFailTimer As Long
    dwInactivityTimeout As Long
    dwSpeakerVolume As Long
    dwSpeakerMode As Long
    dwModemOptions As Long
    dwMaxDTERate As Long
    dwMaxDCERate As Long
    abVariablePortion(1) As Byte
End Type
```

Type **MODEMSETTINGS**
```
    dwActualSize As Long
    dwRequiredSize As Long
    dwDevSpecificOffset As Long
    dwDevSpecificSize As Long
    dwCallSetupFailTimer As Long
    dwInactivityTimeout As Long
    dwSpeakerVolume As Long
    dwSpeakerMode As Long
    dwPreferredModemOptions As Long
    dwNegotiatedModemOptions As Long
    dwNegotiatedDCERate As Long
    abVariablePortion(1) As Byte
End Type
```

Type **MONCBSTRUCT**
```
    cb As Long
    dwTime As Long
    htask As Long
    dwRet As Long
    wType As Long
    wFmt As Long
    hConv As Long
    hsz1 As Long
    hsz2 As Long
    hData As Long
    dwData1 As Long
    dwData2 As Long
    cc As CONVCONTEXT
    cbData As Long
    Data(8) As Long
End Type
```

Type **MONCONVSTRUCT**
```
    cb As Long
    fConnect As Boolean
    dwTime As Long
    htask As Long
    hszSvc As Long
    hszTopic As Long
    hConvClient As Long
    hConvServer As Long
End Type
```

Type **MONERRSTRUCT**
```
    cb As Long
    wLastError As Long
    dwTime As Long
    htask As Long
End Type
```

Type **MONHSZSTRUCT**
```
    cb As Long
    fsAction As Boolean
    dwTime As Long
    hsz As Long
    htask As Long
    str As Byte
End Type
```

Type **MONITOR_INFO_1**
```
    pName As String
End Type
```

Type **MONITOR_INFO_2**
```
    pName As String
    pEnvironment As String
    pDLLName As String
End Type
```

Type **MONLINKSTRUCT**
```
    cb As Long
    dwTime As Long
    htask As Long
    fEstablished As Boolean
    fNoData As Boolean
    hszSvc As Long
    hszTopic As Long
    hszItem As Long
    wFmt As Long
    fServer As Boolean
    hConvServer As Long
    hConvClient As Long
End Type
```

```
Type MONMSGSTRUCT
    cb As Long
    hwndTo As Long
    dwTime As Long
    htask As Long
    wMsg As Long
    wParam As Long
    lParam As Long
    dmhd As DDEML_MSG
                        _HOOK_DATA
End Type

Type MOUSE_EVENT_RECORD
    dwMousePosition As COORD
    dwButtonState As Long
    dwControlKeyState As Long
    dwEventFlags As Long
End Type

Type MOUSEHOOKSTRUCT
    pt As POINTAPI
    hwnd As Long
    wHitTestCode As Long
    dwExtraInfo As Long
End Type

Type MSG
    hwnd As Long
    message As Long
    wParam As Long
    lParam As Long
    time As Long
    pt As POINTAPI
End Type

Type MSGBOXPARAMS
    cbSize As Long
    hwndOwner As Long
    hInstance As Long
    lpszText As String
    lpszCaption As String
    dwStyle As Long
    lpszIcon As String
    dwContextHelpId As Long
    lpfnMsgBoxCallback As Long
    dwLanguageId As Long
End Type

Type MULTIKEYHELP
    mkSize As Long
    mkKeylist As Byte
    szKeyphrase As String*253
End Type

Type NAME_BUFFER
    name As String*NCBNAMSZ
    name_num As Integer
    name_flags As Integer
End Type

Type NCB
    ncb_command As Integer
    ncb_retcode As Integer
    ncb_lsn As Integer
    ncb_num As Integer
    ncb_buffer As String
    ncb_length As Integer
    ncb_callname As String*NCBNAMSZ
    ncb_name As String*NCBNAMSZ
    ncb_rto As Integer
    ncb_sto As Integer
    ncb_post As Long
    ncb_lana_num As Integer
    ncb_cmd_cplt As Integer
    ncb_reserve(10) As Byte
    ncb_event As Long
End Type

Type NETRESOURCE
    dwScope As Long
    dwType As Long
    dwDisplayType As Long
    dwUsage As Long
    lpLocalName As String
    lpRemoteName As String
    lpComment As String
    lpProvider As String
End Type

Type NEWTEXTMETRIC
    tmHeight As Long
    tmAscent As Long
    tmDescent As Long
    tmInternalLeading As Long
    tmExternalLeading As Long
    tmAveCharWidth As Long
    tmMaxCharWidth As Long
    tmWeight As Long
    tmOverhang As Long
    tmDigitizedAspectX As Long
    tmDigitizedAspectY As Long
    tmFirstChar As Byte
    tmLastChar As Byte
    tmDefaultChar As Byte
    tmBreakChar As Byte
    tmItalic As Byte
    tmUnderlined As Byte
    tmStruckOut As Byte
    tmPitchAndFamily As Byte
    tmCharSet As Byte
    ntmFlags As Long
    ntmSizeEM As Long
    ntmCellHeight As Long
    ntmAveWidth As Long
End Type

Type NEWTEXTMETRICEX
    ntmTm As NEWTEXTMETRIC
    ntmFontSig As FONTSIGNATURE
End Type

Type NMHDR
    hwndFrom As Long
    idfrom As Long
    code As Long
End Type

Type NONCLIENTMETRICS
    cbSize As Long
    iBorderWidth As Long
    iScrollWidth As Long
    iScrollHeight As Long
    iCaptionWidth As Long
    iCaptionHeight As Long
    lfCaptionFont As LOGFONT
    iSMCaptionWidth As Long
    iSMCaptionHeight As Long
    lfSMCaptionFont As LOGFONT
    iMenuWidth As Long
    iMenuHeight As Long
    lfMenuFont As LOGFONT
    lfStatusFont As LOGFONT
    lfMessageFont As LOGFONT
End Type

Type NOTIFYICONDATA
    cbSize As Long
    hwnd As Long
    uID As Long
    uFlags As Long
    uCallbackMessage As Long
    hIcon As Long
    szTip As String*64
End Type

Type NUMBERFMT
    NumDigits As Long
    LeadingZero As Long
    Grouping As Long
    lpDecimalSep As String
```

```
    lpThousandSep As String
    NegativeOrder As Long
End Type
```

Type **OFNOTIFY**

```
    hdr As NMHDR
    lpOFN As OPENFILENAME
    pszFile As String
End Type
```

Type **OFSTRUCT**

```
    cBytes As Byte
    fFixedDisk As Byte
    nErrCode As Integer
    Reserved1 As Integer
    Reserved2 As Integer
    szPathName(OFS_MAXPATH
                    NAME) As Byte
End Type
```

Type **OPENFILENAME**

```
    lStructSize As Long
    hwndOwner As Long
    hInstance As Long
    lpstrFilter As String
    lpstrCustomFilter As String
    nMaxCustFilter As Long
    nFilterIndex As Long
    lpstrFile As String
    nMaxFile As Long
    lpstrFileTitle As String
    nMaxFileTitle As Long
    lpstrInitialDir As String
    lpstrTitle As String
    flags As Long
    nFileOffset As Integer
    nFileExtension As Integer
    lpstrDefExt As String
    lCustData As Long
    lpfnHook As Long
    lpTemplateName As String
End Type
```

Type **OSVERSIONINFO**

```
    dwOSVersionInfoSize As Long
    dwMajorVersion As Long
    dwMinorVersion As Long
    dwBuildNumber As Long
    dwPlatformId As Long
    szCSDVersion As String*128
End Type
```

Type **OUTLINETEXTMETRIC**

```
    otmSize As Long
    otmTextMetrics As TEXTMETRIC
    otmFiller As Byte
    otmPanoseNumber As PANOSE
    otmfsSelection As Long
    otmfsType As Long
    otmsCharSlopeRise As Long
    otmsCharSlopeRun As Long
    otmItalicAngle As Long
    otmEMSquare As Long
    otmAscent As Long
    otmDescent As Long
    otmLineGap As Long
    otmsCapEmHeight As Long
    otmsXHeight As Long
    otmrcFontBox As Rect
    otmMacAscent As Long
    otmMacDescent As Long
    otmMacLineGap As Long
    otmusMinimumPPEM As Long
    otmptSubscriptSize As POINTAPI
    otmptSubscriptOffset As POINTAPI
    otmptSuperscriptSize As POINTAPI
    otmptSuperscriptOffset
                    As POINTAPI
    otmsStrikeoutSize As Long
    otmsStrikeoutPosition As Long
    otmsUnderscorePosition As Long
    otmsUnderscoreSize As Long
    otmpFamilyName As String
    otmpFaceName As String
    otmpStyleName As String
    otmpFullName As String
End Type
```

Type **OUTPUT_DEBUG_STRING_INFO**

```
    lpDebugStringData As String
    fUnicode As Integer
    nDebugStringLength As Integer
End Type
```

Type **OVERLAPPED**

```
    Internal As Long
    InternalHigh As Long
    offset As Long
    OffsetHigh As Long
    hEvent As Long
End Type
```

Type **PAGESETUPDLG**

```
    lStructSize As Long
    hwndOwner As Long
    hDevMode As Long
    hDevNames As Long
    flags As Long
    ptPaperSize As POINTAPI
    rtMinMargin As Rect
    rtMargin As Rect
    hInstance As Long
    lCustData As Long
    lpfnPageSetupHook As Long
    lpfnPagePaintHook As Long
    lpPageSetupTemplateName
                    As String
    hPageSetupTemplate As Long
End Type
```

Type **PAINTSTRUCT**

```
    hdc As Long
    fErase As Boolean
    rcPaint As Rect
    fRestore As Boolean
    fIncUpdate As Boolean
    rgbReserved As Byte
End Type
```

Type **PALETTEENTRY**

```
    peRed As Byte
    peGreen As Byte
    peBlue As Byte
    peFlags As Byte
End Type
```

Type **PANOSE**

```
    ulculture As Long
    bFamilyType As Byte
    bSerifStyle As Byte
    bWeight As Byte
    bProportion As Byte
    bContrast As Byte
    bStrokeVariation As Byte
    bArmStyle As Byte
    bLetterform As Byte
    bMidline As Byte
    bXHeight As Byte
End Type
```

Type **PCMWAVEFORMAT**

```
    wf As WAVEFORMAT
    wBitsPerSample As Integer
End Type
```

Type **PELARRAY**

```
    paXCount As Long
    paYCount As Long
    paXExt As Long
    paYExt As Long
    paRGBs As Integer
End Type
```

```
Type PERF_COUNTER_BLOCK
    ByteLength As Long
End Type

Type PERF_COUNTER_DEFINITION
    ByteLength As Long
    CounterNameTitleIndex As Long
    CounterNameTitle As String
    CounterHelpTitleIndex As Long
    CounterHelpTitle As String
    DefaultScale As Long
    DetailLevel As Long
    CounterType As Long
    CounterSize As Long
    CounterOffset As Long
End Type

Type PERF_DATA_BLOCK
    Signature As String*4
    LittleEndian As Long
    Version As Long
    Revision As Long
    TotalByteLength As Long
    HeaderLength As Long
    NumObjectTypes As Long
    DefaultObject As Long
    SystemTime As SYSTEMTIME
    PerfTime As LARGE_INTEGER
    PerfFreq As LARGE_INTEGER
    PerTime100nSec As
                    LARGE_INTEGER
    SystemNameLength As Long
    SystemNameOffset As Long
End Type

Type PERF_INSTANCE_DEFINITION
    ByteLength As Long
    ParentObjectTitleIndex As Long
    ParentObjectInstance As Long
    UniqueID As Long
    NameOffset As Long
    NameLength As Long
End Type

Type PERF_OBJECT_TYPE
    TotalByteLength As Long
    DefinitionLength As Long
    HeaderLength As Long
    ObjectNameTitleIndex As Long
    ObjectNameTitle As String
    ObjectHelpTitleIndex As Long
    ObjectHelpTitle As String
    DetailLevel As Long
    NumCounters As Long
    DefaultCounter As Long
    NumInstances As Long
    CodePage As Long
    PerfTime As LARGE_INTEGER
    PerfFreq As LARGE_INTEGER
End Type

Type PIXELFORMATDESCRIPTOR
    nSize As Integer
    nVersion As Integer
    dwFlags As Long
    iPixelType As Byte
    cColorBits As Byte
    cRedBits As Byte
    cRedShift As Byte
    cGreenBits As Byte
    cGreenShift As Byte
    cBlueBits As Byte
    cBlueShift As Byte
    cAlphaBits As Byte
    cAlphaShift As Byte
    cAccumBits As Byte
    cAccumRedBits As Byte
    cAccumGreenBits As Byte
    cAccumBlueBits As Byte
    cAccumAlphaBits As Byte
    cDepthBits As Byte
    cStencilBits As Byte
    cAuxBuffers As Byte
    iLayerType As Byte
    bReserved As Byte
    dwLayerMask As Long
    dwVisibleMask As Long
    dwDamageMask As Long
End Type

Type POINTAPI
    x As Long
    y As Long
End Type

Type POINTFX
    x As FIXED
    y As FIXED
End Type

Type POINTL
    x As Long
    y As Long
End Type

Type POINTS
    x As Integer
    y As Integer
End Type

Type POLYTEXT
    x As Long
    y As Long
    n As Long
    lpStr As String
    uiFlags As Long
    rcl As Rect
    pdx As Long
End Type

Type PORT_INFO_1
    pName As String
End Type

Type PORT_INFO_2
    pPortName As String
    pMonitorName As String
    pDescription As String
    fPortType As Long
    Reserved As Long
End Type

Type PRINTDLG
    lStructSize As Long
    hwndOwner As Long
    hDevMode As Long
    hDevNames As Long
    hdc As Long
    flags As Long
    nFromPage As Integer
    nToPage As Integer
    nMinPage As Integer
    nMaxPage As Integer
    nCopies As Integer
    hInstance As Long
    lCustData As Long
    lpfnPrintHook As Long
    lpfnSetupHook As Long
    lpPrintTemplateName As String
    lpSetupTemplateName As String
    hPrintTemplate As Long
    hSetupTemplate As Long
End Type

Type PRINTER_DEFAULTS
    pDatatype As String
    pDevMode As DEVMODE
    DesiredAccess As Long
End Type

Type PRINTER_INFO_1
    flags As Long
    pDescription As String
    pName As String
    pComment As String
End Type
```

```
Type PRINTER_INFO_2
    pServerName As String
    pPrinterName As String
    pShareName As String
    pPortName As String
    pDriverName As String
    pComment As String
    pLocation As String
    pDevMode As DEVMODE
    pSepFile As String
    pPrintProcessor As String
    pDatatype As String
    pParameters As String
    pSecurityDescriptor As SECURITY
                        _DESCRIPTOR
    Attributes As Long
    Priority As Long
    DefaultPriority As Long
    StartTime As Long
    UntilTime As Long
    Status As Long
    cJobs As Long
    AveragePPM As Long
End Type
```

```
Type PRINTER_INFO_3
    pSecurityDescriptor As
            SECURITY_DESCRIPTOR
End Type
```

```
Type PRINTER_INFO_4
    pPrinterName As String
    pServerName As String
    Attributes As Long
End Type
```

```
Type PRINTER_INFO_5
    pPrinterName As String
    pPortName As String
    Attributes As Long
    DeviceNotSelectedTimeout As Long
    TransmissionRetryTimeout As Long
End Type
```

```
Type PRINTPROCESSOR_INFO_1
    pName As String
End Type
```

```
Type PRIVILEGE_SET
    PrivilegeCount As Long
    Control As Long
    Privilege(ANYSIZE_ARRAY) As
    LUID_AND_ATTRIBUTES
End Type
```

```
Type PROCESS_INFORMATION
    hProcess As Long
    hThread As Long
    dwProcessId As Long
    dwThreadId As Long
End Type
```

```
Type PROVIDOR_INFO_1
    pName As String
    pEnvironment As String
    pDLLName As String
End Type
```

```
Type QUERY_SERVICE_CONFIG
    dwServiceType As Long
    dwStartType As Long
    dwErrorControl As Long
    lpBinaryPathName As String
    lpLoadOrderGroup As String
    dwTagId As Long
    lpDependencies As String
    lpServiceStartName As String
    lpDisplayName As String
End Type
```

```
Type UERY_SERVICE_LOCK_STATUS
    fIsLocked As Long
    lpLockOwner As String
    dwLockDuration As Long
End Type
```

```
Type RASTERIZER_STATUS
    nSize As Integer
    wFlags As Integer
    nLanguageID As Integer
End Type
```

```
Type RECT
    Left As Long
    Top As Long
    Right As Long
    Bottom As Long
End Type
```

```
Type RECTL
    Left As Long
    Top As Long
    Right As Long
    Bottom As Long
End Type
```

```
Type RGBQUAD
    rgbBlue As Byte
    rgbGreen As Byte
    rgbRed As Byte
    rgbReserved As Byte
End Type
```

```
Type RGBTRIPLE
    rgbtBlue As Byte
    rgbtGreen As Byte
    rgbtRed As Byte
End Type
```

```
Type RGNDATA
    rdh As RGNDATAHEADER
    Buffer As Byte
End Type
```

```
Type RGNDATAHEADER
    dwSize As Long
    iType As Long
    nCount As Long
    nRgnSize As Long
    rcBound As Rect
End Type
```

```
Type RIP_INFO
    dwError As Long
    dwType As Long
End Type
```

```
Type SCROLLINFO
    cbSize As Long
    fMask As Long
    nMin As Long
    nMax As Long
    nPage As Long
    nPos As Long
    nTrackPos As Long
End Type
```

```
Type SECURITY_ATTRIBUTES
    nLength As Long
    lpSecurityDescriptor As Long
    bInheritHandle As Boolean
End Type
```

```
Type SECURITY_DESCRIPTOR
    Revision As Byte
    Sbz1 As Byte
    Control As Long
    Owner As Long
    Group As Long
    Sacl As ACL
    Dacl As ACL
End Type
```

```
Type SECURITY_QUALITY
                    _OF_SERVICE
    Length As Long
    Impersonationlevel As Integer
    ContextTrackingMode As Integer
    EffectiveOnly As Boolean
End Type

Type SERVICE_STATUS
    dwServiceType As Long
    dwCurrentState As Long
    dwControlsAccepted As Long
    dwWin32ExitCode As Long
    dwServiceSpecificExitCode
                        As Long
    dwCheckPoint As Long
    dwWaitHint As Long
End Type

Type SERVICE_TABLE_ENTRY
    lpServiceName As String
    lpServiceProc As Long
End Type

Type SESSION_BUFFER
    lsn As Integer
    State As Integer
    local_name As String*NCBNAMSZ
    remote_name As
                String*NCBNAMSZ
    rcvs_outstanding As Integer
    sends_outstanding As Integer
End Type

Type SESSION_HEADER
    sess_name As Integer
    num_sess As Integer
    rcv_dg_outstanding As Integer
    rcv_any_outstanding As Integer
End Type

Type SHELLEXECUTEINFO
    cbSize As Long
    fMask As Long
    hwnd As Long
    lpVerb As String
    lpFile As String
    lpParameters As String
    lpDirectory As String
    nShow As Long
    hInstApp As Long
    lpIDList As Long
    lpClass As String
    hkeyClass As Long
    dwHotKey As Long
    hIcon As Long
    hProcess As Long
End Type

Type SHFILEINFO
    hIcon As Long
    iIcon As Long
    dwAttributes As Long
    szDisplayName As
String*MAX_PATH
    szTypeName As String*80
End Type

Type SHFILEOPSTRUCT
    hwnd As Long
    wFunc As Long
    pFrom As String
    pTo As String
    fFlags As Integer
    fAnyOperationsAborted As Boolean
    hNameMappings As Long
    lpszProgressTitle As String
End Type

Type SHNAMEMAPPING
    pszOldPath As String
    pszNewPath As String
    cchOldPath As Long
    cchNewPath As Long
End Type

Type SID_AND_ATTRIBUTES
    Sid As Long
    Attributes As Long
End Type

Type SID_IDENTIFIER_AUTHORITY
    Value(6) As Byte
End Type

Type Size
    cx As Long
    cy As Long
End Type

Type SIZEL
    cx As Long
    cy As Long
End Type

Type SMALL_RECT
    Left As Integer
    Top As Integer
    Right As Integer
    Bottom As Integer
End Type

Type smpte
    hour As Byte
    min As Byte
    sec As Byte
    frame As Byte
    fps As Byte
    dummy As Byte
    pad(2) As Byte
End Type

Type STARTUPINFO
    cb As Long
    lpReserved As String
    lpDesktop As String
    lpTitle As String
    dwX As Long
    dwY As Long
    dwXSize As Long
    dwYSize As Long
    dwXCountChars As Long
    dwYCountChars As Long
    dwFillAttribute As Long
    dwFlags As Long
    wShowWindow As Integer
    cbReserved2 As Integer
    lpReserved2 As Byte
    hStdInput As Long
    hStdOutput As Long
    hStdError As Long
End Type

Type STYLEBUF
    dwStyle As Long
    szDescription As String*STYLE
                _DESCRIPTION_SIZE
End Type

Type SYSTEM_ALARM_ACE
    Header As ACE_HEADER
    Mask As Long
    SidStart As Long
End Type

Type SYSTEM_AUDIT_ACE
    Header As ACE_HEADER
    Mask As Long
    SidStart As Long
End Type
```

```
Type SYSTEM_INFO
    dwOemID As Long
    dwPageSize As Long
    lpMinimumApplicationAddress As Long
    lpMaximumApplicationAddress As Long
    dwActiveProcessorMask As Long
    dwNumberOrfProcessors As Long
    dwProcessorType As Long
    dwAllocationGranularity As Long
    dwReserved As Long
End Type
```

```
Type SYSTEM_POWER_STATUS
    ACLineStatus As Byte
    BatteryFlag As Byte
    BatteryLifePercent As Byte
    Reserved1 As Byte
    BatteryLifeTime As Long
    BatteryFullLifeTime As Long
End Type
```

```
Type SYSTEMTIME
    wYear As Integer
    wMonth As Integer
    wDayOfWeek As Integer
    wDay As Integer
    wHour As Integer
    wMinute As Integer
    wSecond As Integer
    wMilliseconds As Integer
End Type
```

```
Type Target
    dwType As Long
    dwDeviceID As Long
    wMid As Integer
    wPid As Integer
    vDriverVersion As Long
    szPname As String*MAXPNAMELEN
End Type
```

```
Type TEXTMETRIC
    tmHeight As Long
    tmAscent As Long
    tmDescent As Long
    tmInternalLeading As Long
    tmExternalLeading As Long
    tmAveCharWidth As Long
    tmMaxCharWidth As Long
    tmWeight As Long
    tmOverhang As Long
    tmDigitizedAspectX As Long
    tmDigitizedAspectY As Long
    tmFirstChar As Byte
    tmLastChar As Byte
    tmDefaultChar As Byte
    tmBreakChar As Byte
    tmItalic As Byte
    tmUnderlined As Byte
    tmStruckOut As Byte
    tmPitchAndFamily As Byte
    tmCharSet As Byte
End Type
```

```
Type TIME_ZONE_INFORMATION
    Bias As Long
    StandardName(32) As Integer
    StandardDate As SYSTEMTIME
    StandardBias As Long
    DaylightName(32) As Integer
    DaylightDate As SYSTEMTIME
    DaylightBias As Long
End Type
```

```
Type TIMECAPS
    wPeriodMin As Long
    wPeriodMax As Long
End Type
```

```
Type TOKEN_GROUPS
    GroupCount As Long
    Groups(ANYSIZE_ARRAY) As
            SID_AND_ATTRIBUTES
End Type
```

```
Type TOKEN_PRIVILEGES
    PrivilegeCount As Long
    Privileges(ANYSIZE_ARRAY) As
            LUID_AND_ATTRIBUTES
End Type
```

```
Type TPMPARAMS
    cbSize As Long
    rcExclude As Rect
End Type
```

```
Type TTPOLYCURVE
    wType As Integer
    cpfx As Integer
    apfx As POINTFX
End Type
```

```
Type TTPOLYGONHEADER
    cb As Long
    dwType As Long
    pfxStart As POINTFX
End Type
```

```
Type UNLOAD_DLL_DEBUG_INFO
    lpBaseOfDll As Long
End Type
```

```
Type VS_FIXEDFILEINFO
    dwSignature As Long
    dwStrucVersion As Long
    dwFileVersionMS As Long
    dwFileVersionLS As Long
    dwProductVersionMS As Long
    dwProductVersionLS As Long
    dwFileFlagsMask As Long
    dwFileFlags As Long
    dwFileOS As Long
    dwFileType As Long
    dwFileSubtype As Long
    dwFileDateMS As Long
    dwFileDateLS As Long
End Type
```

```
Type WAVEFORMAT
    wFormatTag As Integer
    nChannels As Integer
    nSamplesPerSec As Long
    nAvgBytesPerSec As Long
    nBlockAlign As Integer
End Type
```

```
Type WAVEHDR
    lpData As String
    dwBufferLength As Long
    dwBytesRecorded As Long
    dwUser As Long
    dwFlags As Long
    dwLoops As Long
    lpNext As Long
    Reserved As Long
End Type
```

```
Type WAVEINCAPS
    wMid As Integer
    wPid As Integer
    vDriverVersion As Long
    szPname As
            String*MAXPNAMELEN
    dwFormats As Long
    wChannels As Integer
End Type
```

```
Type WAVEOUTCAPS
    wMid As Integer
    wPid As Integer
    vDriverVersion As Long
```

```
    szPname As
            String*MAXPNAMELEN
    dwFormats As Long
    wChannels As Integer
    dwSupport As Long
End Type
```

```
Type WIN32_FIND_DATA
    dwFileAttributes As Long
    ftCreationTime As FILETIME
    ftLastAccessTime As FILETIME
    ftLastWriteTime As FILETIME
    nFileSizeHigh As Long
    nFileSizeLow As Long
    dwReserved0 As Long
    dwReserved1 As Long
    cFileName As String*MAX_PATH
    cAlternate As String*14
End Type
```

```
Type WIN32_STREAM_ID
    dwStreamID As Long
    dwStreamAttributes As Long
    dwStreamSizeLow As Long
    dwStreamSizeHigh As Long
    dwStreamNameSize As Long
    cStreamName As Byte
End Type
```

```
Type WINDOW_BUFFER_SIZE
                    _RECORD
    dwSize As COORD
End Type
```

```
Type WINDOWPLACEMENT
    Length As Long
    flags As Long
    showCmd As Long
    ptMinPosition As POINTAPI
    ptMaxPosition As POINTAPI
    rcNormalPosition As Rect
End Type
```

```
Type WINDOWPOS
    hwnd As Long
    hWndInsertAfter As Long
    x As Long
    y As Long
    cx As Long
    cy As Long
    flags As Long
End Type
```

```
Type WNDCLASS
    style As Long
    lpfnwndproc As Long
    cbClsextra As Long
    cbWndExtra2 As Long
    hInstance As Long
    hIcon As Long
    hCursor As Long
    hbrBackground As Long
    lpszMenuName As String
    lpszClassName As String
End Type
```

```
Type WNDCLASSEX
    cbSize As Long
    style As Long
    lpfnWndProc As Long
    cbClsExtra As Long
    cbWndExtra As Long
    hInstance As Long
    hIcon As Long
    hCursor As Long
    hbrBackground As Long
    lpszMenuName As String
    lpszClassName As String
    hIconSm As Long
End Type
```

```
Type XFORM
    eM11 As Double
    eM12 As Double
    eM21 As Double
    eM22 As Double
    eDx As Double
    eDy As Double
End Type
```

付録D　Visual Basic用定数一覧

以下は、本書で扱っているAPI関数が利用する定数をアルファベット順に列挙した一覧である。コーディングする際には、Constステートメントに続けて関数の定義をすること。メッセージに関する定数は2つめの表にまとめてある。利用方法やコーディングについては「1.5　Visual Basicからの Windows API呼び出し」を参照してもらいたい。Visual C++では、ヘッダファイルをインクルードすることで定義されるので特に利用する必要はない。

●Win32定数一覧

AC_LINE_BACKUP_POWER = &H2
AC_LINE_OFFLINE = &H0
AC_LINE_ONLINE = &H1
AC_LINE_UNKNOWN = &HFF
BACKUP_ALTERNATE_DATA = &H4
BACKUP_DATA = &H1
BACKUP_EA_DATA = &H2
BACKUP_LINK = &H5
BACKUP_SECURITY_DATA = &H3
BATTERY_FLAG_CHARGING = &H8
BATTERY_FLAG_CRITICAL = &H4
BATTERY_FLAG_HIGH = &H1
BATTERY_FLAG_LOW = &H2
BATTERY_FLAG_NO_BATTERY = &H80
BATTERY_FLAG_UNKNOWN = &HFF
BATTERY_LIFE_UNKNOWN = &HFFFF
BATTERY_PERCENTAGE_UNKNOWN =&HFF
BAUD_075 = &H1&
BAUD_110 = &H2&
BAUD_115200 = &H20000
BAUD_1200 = &H40&
BAUD_128K = &H10000
BAUD_134_5 = &H4&
BAUD_14400 = &H1000&
BAUD_150 = &H8&
BAUD_1800 = &H80&
BAUD_19200 = &H2000&
BAUD_2400 = &H100&
BAUD_300 = &H10&
BAUD_38400 = &H4000&
BAUD_4800 = &H200&
BAUD_56K = &H8000&
BAUD_57600 = &H40000
BAUD_600 = &H20&
BAUD_7200 = &H400&
BAUD_9600 = &H800&
BAUD_USER = &H10000000
CBR_110 = 110
CBR_115200 = 115200
CBR_1200 = 1200
CBR_128000 = 128000
CBR_14400 = 14400
CBR_19200 = 19200
CBR_2400 = 2400
CBR_256000 = 256000
CBR_300 = 300
CBR_38400 = 38400
CBR_4800 = 4800
CBR_56000 = 56000
CBR_57600 = 57600
CBR_600 = 600
CBR_9600 = 9600
CBR_BLOCK = &HFFFF
CBS_AUTOHSCROLL = &H40&
CBS_DISABLENOSCROLL = &H800&
CBS_DROPDOWN = &H2&
CBS_DROPDOWNLIST = &H3&
CBS_HASSTRINGS = &H200&
CBS_NOINTEGRALHEIGHT = &H400&
CBS_OEMCONVERT = &H80&
CBS_OWNERDRAWFIXED = &H10&
CBS_OWNERDRAWVARIABLE = &H20&
CBS_SIMPLE = &H1&
CBS_SORT = &H100&
CE_BREAK = &H10
CE_DNS = &H800
CE_FRAME = &H8
CE_IOE = &H400
CE_MODE = &H8000
CE_OOP = &H1000

CE_OVERRUN = &H2
CE_PTO = &H200
CE_RXOVER = &H1
CE_RXPARITY = &H4
CE_TXFULL = &H100
CLR_INVALID = &HFFFF
CLRBREAK = 9
CLRDTR = 6
CLRRTS = 4
CREATE_ALWAYS = 2
CREATE_NEW = 1
CREATE_NEW_CONSOLE = &H10
CREATE_NEW_PROCESS_GROUP =&H200
CREATE_NO_WINDOW = &H8000000
CREATE_PROCESS_DEBUG_EVENT = 3
CREATE_SUSPENDED = &H4
CREATE_THREAD_DEBUG_EVENT = 2
DATABITS_16 = &H10&
DATABITS_16X = &H20&
DATABITS_5 = &H1&
DATABITS_6 = &H2&
DATABITS_7 = &H4&
DATABITS_8 = &H8&
DDD_EXACT_MATCH_ON_REMOVE = &H4
DDD_RAW_TARGET_PATH = &H1
DDD_REMOVE_DEFINITION = &H2
DEBUG_ONLY_THIS_PROCESS = &H2
DEBUG_PROCESS = &H1
DETACHED_PROCESS = &H8
DRIVE_CDROM = 5
DRIVE_FIXED = 3
DRIVE_RAMDISK = 6
DRIVE_REMOTE = 4
DRIVE_REMOVABLE = 2
DRIVERVERSION = 0
DRV_CANCEL = DRVCNF_CANCEL
DRV_CLOSE = &H4
DRV_CONFIGURE = &H7
DRV_DISABLE = &H5
DRV_ENABLE = &H2
DRV_EXITSESSION = &HB
DRV_FREE = &H6
DRV_INSTALL = &H9
DRV_LOAD = &H1
DRV_MCI_FIRST = DRV_RESERVED
DRV_MCI_LAST = DRV_RESERVED + &HFFF
DRV_OK = DRVCNF_OK
DRV_OPEN = &H3
DRV_POWER = &HF
DRV_QUERYCONFIGURE = &H8
DRV_REMOVE = &HA
DRV_RESERVED = &H800
DRV_RESTART = DRVCNF_RESTART
DRV_USER = &H4000
DTR_CONTROL_DISABLE = &H0
DTR_CONTROL_ENABLE = &H1
DTR_CONTROL_HANDSHAKE = &H2
EV_BREAK = &H40
EV_CTS = &H8
EV_DSR = &H10
EV_ERR = &H80
EV_EVENT1 = &H800
EV_EVENT2 = &H1000
EV_PERR = &H200
EV_RING = &H100
EV_RLSD = &H20
EV_RX80FULL = &H400
EV_RXCHAR = &H1
EV_RXFLAG = &H2
EV_TXEMPTY = &H4
EVENPARITY = 2
EXCEPTION_CONTINUE_EXECUTION = -1
EXCEPTION_CONTINUE_SEARCH = 0
EXCEPTION_DEBUG_EVENT = 1
EXCEPTION_EXECUTE_HANDLER = 1
EXCEPTION_MAXIMUM_PARAMETERS = 15
EXIT_PROCESS_DEBUG_EVENT = 5
EXIT_THREAD_DEBUG_EVENT = 4
EXT_DEVICE_CAPS = 4099
FILE_ADD_FILE = (&H2)
FILE_ADD_SUBDIRECTORY = (&H4)
FILE_ALL_ACCESS=(STANDARD_RIGHTS_
REQUIRED Or SYNCHRONIZE Or &H1FF)
FILE_APPEND_DATA = (&H4)
FILE_ATTRIBUTE_ARCHIVE = &H20
FILE_ATTRIBUTE_COMPRESSED= &H800
FILE_ATTRIBUTE_DIRECTORY = &H10
FILE_ATTRIBUTE_HIDDEN = &H2
FILE_ATTRIBUTE_NORMAL = &H80
FILE_ATTRIBUTE_READONLY = &H1

FILE_ATTRIBUTE_SYSTEM = &H4
FILE_ATTRIBUTE_TEMPORARY = &H100
FILE_BEGIN = 0
FILE_CASE_PRESERVED_NAMES = &H2
FILE_CASE_SENSITIVE_SEARCH = &H1
FILE_CREATE_PIPE_INSTANCE = (&H4)
FILE_CURRENT = 1
FILE_DELETE_CHILD = (&H40)
FILE_END = 2
FILE_EXECUTE = (&H20)
FILE_FILE_COMPRESSION = &H10
FILE_FLAG_BACKUP_SEMANTICS = &H2000000
FILE_FLAG_DELETE_ON_CLOSE= &H4000000
FILE_FLAG_NO_BUFFERING = &H20000000
FILE_FLAG_OVERLAPPED = &H40000000
FILE_FLAG_POSIX_SEMANTICS = &H1000000
FILE_FLAG_RANDOM_ACCESS = &H10000000
FILE_FLAG_SEQUENTIAL_SCAN = &H8000000
FILE_FLAG_WRITE_THROUGH = &H80000000
FILE_LIST_DIRECTORY = (&H1)
FILE_MAP_ALL_ACCESS = SECTION_ALL_ACCESS
FILE_MAP_COPY = SECTION_QUERY
FILE_MAP_READ = SECTION_MAP_READ
FILE_MAP_WRITE = SECTION_MAP_WRITE
FILE_NOTIFY_CHANGE_ATTRIBUTES = &H4
FILE_NOTIFY_CHANGE_DIR_NAME = &H2
FILE_NOTIFY_CHANGE_FILE_NAME = &H1
FILE_NOTIFY_CHANGE_LAST_WRITE = &H10
FILE_NOTIFY_CHANGE_SECURITY = &H100
FILE_NOTIFY_CHANGE_SIZE = &H8
FILE_PERSISTENT_ACLS = &H8
FILE_READ_ATTRIBUTES = (&H80)
FILE_READ_DATA = (&H1)
FILE_READ_EA = (&H8)
FILE_READ_PROPERTIES = FILE_READ_EA
FILE_SHARE_READ = &H1
FILE_SHARE_WRITE = &H2
FILE_TRAVERSE = (&H20)
FILE_TYPE_CHAR = &H2
FILE_TYPE_DISK = &H1
FILE_TYPE_PIPE = &H3
FILE_TYPE_REMOTE = &H8000
FILE_TYPE_UNKNOWN = &H0
FILE_UNICODE_ON_DISK = &H4
FILE_VOLUME_IS_COMPRESSED = &H8000
FILE_WRITE_ATTRIBUTES = (&H100)
FILE_WRITE_DATA = (&H2)
FILE_WRITE_EA = (&H10)
FILE_WRITE_PROPERTIES = FILE_WRITE_EA
FORMAT_MESSAGE_ALLOCATE_BUFFER = &H100
FORMAT_MESSAGE_ARGUMENT_ARRAY = &H2000
FORMAT_MESSAGE_FROM_HMODULE = &H800
FORMAT_MESSAGE_FROM_STRING = &H400
FORMAT_MESSAGE_FROM_SYSTEM = &H1000
FORMAT_MESSAGE_IGNORE_INSERTS = &H200
FORMAT_MESSAGE_MAX_WIDTH_MASK = &HFF
GET_TAPE_DRIVE_INFORMATION = 1
GET_TAPE_MEDIA_INFORMATION = 0
GETCOLORTABLE = 5
GETDEVICEUNITS = 42
GETEXTENDEDTEXTMETRICS = 256
GETEXTENTTABLE = 257
GETFACENAME = 513
GETPAIRKERNTABLE = 258
GETPENWIDTH = 16
GETPHYSPAGESIZE = 12
GETPRINTINGOFFSET = 13
GETSCALINGFACTOR = 14
GETSETPAPERBINS = 29
GETSETPAPERMETRICS = 35
GETSETPRINTORIENT = 30
GETSETSCREENPARAMS = 3072
GETTECHNOLGY = 20
GETTECHNOLOGY = 20
GETTRACKKERNTABLE = 259
GETVECTORBRUSHSIZE = 27
GETVECTORPENSIZE = 26
GMEM_DDESHARE = &H2000
GMEM_DISCARDABLE = &H100
GMEM_DISCARDED = &H4000
GMEM_FIXED = &H0
GMEM_INVALID_HANDLE = &H8000
GMEM_LOCKCOUNT = &HFF
GMEM_LOWER = GMEM_NOT_BANKED
GMEM_MODIFY = &H80
GMEM_MOVEABLE = &H2
GMEM_NOCOMPACT = &H10
GMEM_NODISCARD = &H20
GMEM_NOT_BANKED = &H1000
GMEM_NOTIFY = &H4000

GMEM_SHARE = &H2000
GMEM_VALID_FLAGS = &H7F72
GMEM_ZEROINIT = &H40
IDI_APPLICATION = 32512&
IDI_ASTERISK = 32516&
IDI_EXCLAMATION = 32515&
IDI_HAND = 32513&
IDI_QUESTION = 32514&
IDIGNORE = 5
IDLE_PRIORITY_CLASS = &H40
IDNO = 7
IDOK = 1
IDRETRY = 4
IDYES = 6
IE_BADID = (-1)
IE_BAUDRATE = (-12)
IE_BYTESIZE = (-11)
IE_DEFAULT = (-5)
IE_HARDWARE = (-10)
IE_MEMORY = (-4)
IE_NOPEN = (-3)
IE_OPEN = (-2)
IGNORE = 0
LMEM_DISCARDABLE = &HF00
LMEM_DISCARDED = &H4000
LMEM_FIXED = &H0
LMEM_INVALID_HANDLE = &H8000
LMEM_LOCKCOUNT = &HFF
LMEM_MODIFY = &H80
LMEM_MOVEABLE = &H2
LMEM_NOCOMPACT = &H10
LMEM_NODISCARD = &H20
LMEM_VALID_FLAGS = &HF72
LMEM_ZEROINIT = &H40
LOAD_DLL_DEBUG_EVENT = 6
LOCKFILE_EXCLUSIVE_LOCK = &H2
LOCKFILE_FAIL_IMMEDIATELY = &H1
LOGON32_LOGON_BATCH = 4
LOGON32_LOGON_INTERACTIVE = 2
LOGON32_LOGON_SERVICE = 5
LOGON32_PROVIDER_DEFAULT = 0
LOGON32_PROVIDER_WINNT35 = 1
LOGPIXELSX = 88
LOGPIXELSY = 90
LPTR = (LMEM_FIXED + LMEM_ZEROINIT)
LPTx = &H80
MARKPARITY = 3
MAX_DEFAULTCHAR = 2
MAX_LANA = 254
MAX_LEADBYTES = 12
MAX_MONITORS = 4
MAX_PATH = 32
MAX_PRIORITY = 99
MAXByte = &HFF
MAXCHAR = &H7F
MAXDWORD = &HFFFF
MAXERRORLENGTH = 128
MAXIMUM_ALLOWED = &H2000000
MAXLONG = &H7FFFFFFF
MAXPNAMELEN = 32
MAXSHORT = &H7FFF
MAXSTRETCHBLTMODE = 4
MAXWORD = &HFFFF
MOVEFILE_COPY_ALLOWED = &H2
MOVEFILE_DELAY_UNTIL_REBOOT = &H4
MOVEFILE_REPLACE_EXISTING = &H1
MS_CTS_ON = &H10&
MS_DSR_ON = &H20&
MS_NBF = "MNBF"
MS_RING_ON = &H40&
MS_RLSD_ON = &H80&
MSGF_DDEMGR = &H8001
MSGF_DIALOGBOX = 0
MSGF_MAINLOOP = 8
MSGF_MAX = 8
MSGF_MENU = 2
MSGF_MESSAGEBOX = 1
MSGF_MOVE = 3
MSGF_NEXTWINDOW = 6
MSGF_SCROLLBAR = 5
MSGF_SIZE = 4
MSGF_USER = 4096
NMPWAIT_NOWAIT = &H1
NMPWAIT_USE_DEFAULT_WAIT = &H0
NMPWAIT_WAIT_FOREVER = &HFFFF
NORMAL_PRIORITY_CLASS = &H20
ODDPARITY = 1
OF_CANCEL = &H800
OF_CREATE = &H1000
OF_DELETE = &H200

OF_EXIST = &H4000
OF_PARSE = &H100
OF_PROMPT = &H2000
OF_READ = &H0
OF_READWRITE = &H2
OF_REOPEN = &H8000
OF_SHARE_COMPAT = &H0
OF_SHARE_DENY_NONE = &H40
OF_SHARE_DENY_READ = &H30
OF_SHARE_DENY_WRITE = &H20
OF_SHARE_EXCLUSIVE = &H10
OF_VERIFY = &H400
OF_WRITE = &H1
OFN_ALLOWMULTISELECT = &H200
OFN_CREATEPROMPT = &H2000
OFN_ENABLEHOOK = &H20
OFN_ENABLETEMPLATE = &H40
OFN_ENABLETEMPLATEHANDLE = &H80
OFN_EXPLORER = &H80000
OFN_EXTENSIONDIFFERENT = &H400
OFN_FILEMUSTEXIST = &H1000
OFN_HIDEREADONLY = &H4
OFN_LONGNAMES = &H200000
OFN_NOCHANGEDIR = &H8
OFN_NODEREFERENCELINKS = &H100000
OFN_NOLONGNAMES = &H40000
OFN_NONETWORKBUTTON = &H20000
OFN_NOREADONLYRETURN = &H8000
OFN_NOTESTFILECREATE = &H10000
OFN_NOVALIDATE = &H100
OFN_OVERWRITEPROMPT = &H2
OFN_PATHMUSTEXIST = &H800
OFN_READONLY = &H1
OFN_SHAREAWARE = &H4000
OFN_SHAREFALLTHROUGH = 2
OFN_SHARENOWARN = 1
OFN_SHAREWARN = 0
OFN_SHOWHELP = &H10
OFS_MAXPATHNAME = 128
OPEN_ALWAYS = 4
OPEN_EXISTING = 3
OPENCHANNEL = 4110
PARITY_EVEN = &H400&
PARITY_MARK = &H800&
PARITY_NONE = &H100&
PARITY_ODD = &H200&
PARITY_SPACE = &H1000&
PCF_16BITMODE = &H200&
PCF_DTRDSR = &H1&
PCF_INTTIMEOUTS = &H80&
PCF_PARITY_CHECK = &H8&
PCF_RLSD = &H4&
PCF_RTSCTS = &H2&
PCF_SETXCHAR = &H20&
PCF_SPECIALCHARS = &H100&
PCF_TOTALTIMEOUTS = &H40&
PCF_XONXOFF = &H10&
PIPE_ACCESS_DUPLEX = &H3
PIPE_ACCESS_INBOUND = &H1
PIPE_ACCESS_OUTBOUND = &H2
PIPE_CLIENT_END = &H0
PIPE_NOWAIT = &H1
PIPE_READMODE_BYTE = &H0
PIPE_READMODE_MESSAGE = &H2
PIPE_SERVER_END = &H1
PIPE_TYPE_BYTE = &H0
PIPE_TYPE_MESSAGE = &H4
PIPE_UNLIMITED_INSTANCES = 255
PIPE_WAIT = &H0
PROCESS_HEAP_ENTRY_BUSY = &H4
PROCESS_HEAP_ENTRY_DDESHARE = &H20
PROCESS_HEAP_ENTRY_MOVEABLE = &H10
PROCESS_HEAP_REGION = &H1
PROCESS_HEAP_UNCOMMITTED_RANGE = &H2
PROFILE_KERNEL = &H20000000
PROFILE_SERVER = &H40000000
PROFILE_USER = &H10000000
PROOF_QUALITY = 2
PST_FAX = &H21&
PST_LAT = &H101&
PST_NETWORK_BRIDGE = &H100&
PST_PARALLELPORT = &H2&
PST_RS232 = &H1&
PST_RS422 = &H3&
PST_RS423 = &H4&
PST_RS449 = &H5&
PST_SCANNER = &H22&
PST_TCPIP_TELNET = &H102&
PST_UNSPECIFIED = &H0&
PST_X25 = &H103&

PURGE_RXABORT = &H2
PURGE_RXCLEAR = &H8
PURGE_TXABORT = &H1
PURGE_TXCLEAR = &H4
REALTIME_PRIORITY_CLASS = &H100
RESETDEV = 7
RTS_CONTROL_DISABLE = &H0
RTS_CONTROL_ENABLE = &H1
RTS_CONTROL_HANDSHAKE = &H2
RTS_CONTROL_TOGGLE = &H3
S_ALLTHRESHOLD = 2
S_FALSE = &H1
S_LEGATO = 1
S_NORMAL = 0
S_OK = &H0
S_PERIOD1024 = 1
S_PERIOD2048 = 2
S_PERIOD512 = 0
S_PERIODVOICE = 3
S_QUEUEEMPTY = 0
S_SERBDNT = (-5)
S_SERDCC = (-7)
S_SERDDR = (-14)
S_SERDFQ = (-13)
S_SERDLN = (-6)
S_SERDMD = (-10)
S_SERDPT = (-12)
S_SERDSH = (-11)
S_SERDSR = (-15)
S_SERDST = (-16)
S_SERDTP = (-8)
S_SERDVL = (-9)
S_SERDVNA = (-1)
S_SERMACT = (-3)
S_SEROFM = (-2)
S_SERQFUL = (-4)
S_STACCATO = 2
S_THRESHOLD = 1
S_WHITE1024 = 5
S_WHITE2048 = 6
S_WHITE512 = 4
S_WHITEVOICE = 7
SCS_32BIT_BINARY = 0
SCS_CAP_COMPSTR = &H1
SCS_CAP_MAKEREAD = &H2
SCS_CHANGEATTR = (GCS_COMPREADATTR
Or GCS_COMPATTR)
SCS_CHANGECLAUSE = (GCS_
COMPREADCLAUSE Or GCS_COMPCLAUSE)
SCS_DOS_BINARY = 1
SCS_OS216_BINARY = 5
SCS_PIF_BINARY = 3
SCS_POSIX_BINARY = 4
SCS_SETSTR = (GCS_COMPREADSTR
Or GCS_COMPSTR)
SCS_WOW_BINARY = 2
SECURITY_ANONYMOUS_LOGON_RID = &H7
SECURITY_BATCH_RID = &H3
SECURITY_BUILTIN_DOMAIN_RID = &H20
SECURITY_CONTEXT_TRACKING = &H40000
SECURITY_CREATOR_GROUP_RID = &H1
SECURITY_CREATOR_OWNER_RID = &H0
SECURITY_DESCRIPTOR_MIN_LENGTH = (20)
SECURITY_DESCRIPTOR_REVISION = (1)
SECURITY_DESCRIPTOR_REVISION1 = (1)
SECURITY_DIALUP_RID = &H1
SECURITY_EFFECTIVE_ONLY = &H80000
SECURITY_INTERACTIVE_RID = &H4
SECURITY_LOCAL_RID = &H0
SECURITY_LOCAL_SYSTEM_RID = &H12
SECURITY_LOGON_IDS_RID = &H5
SECURITY_NETWORK_RID = &H2
SECURITY_NT_NON_UNIQUE = &H15
SECURITY_NULL_RID = &H0
SECURITY_SERVICE_RID = &H6
SECURITY_SQOS_PRESENT = &H100000
SECURITY_VALID_SQOS_FLAGS = &H1F0000
SECURITY_WORLD_RID = &H0
SEM_FAILCRITICALERRORS = &H1
SEM_NOGPFAULTERRORBOX = &H2
SEM_NOOPENFILEERRORBOX = &H8000
SET_ARC_DIRECTION = 4102
SET_BACKGROUND_COLOR = 4103
SET_BOUNDS = 4109
SET_CLIP_BOX = 4108
SET_MIRROR_MODE = 4110
SET_POLY_MODE = 4104
SET_SCREEN_ANGLE = 4105
SET_SPREAD = 4106
SET_TAPE_DRIVE_INFORMATION = 1

SET_TAPE_MEDIA_INFORMATION = 0
SETBREAK = 8
SETCHARSET = 772
SETCOLORTABLE = 4
SETCOPYCOUNT = 17
SETDIBSCALING = 32
SETDTR = 5
SETKERNTRACK = 770
SETLINECAP = 21
SETLINEJOIN = 22
SetMiterLimitC = 23
SETRGBSTRING = "commdlg_SetRGBColor"
SETRTS = 3
SETXOFF = 1
SETXON = 2
SP_APPABORT = (-2)
SP_BAUD = &H2&
SP_DATABITS = &H4&
SP_ERROR = (-1)
SP_HANDSHAKING = &H10&
SP_NOTREPORTED = &H4000
SP_OUTOFDISK = (-4)
SP_OUTOFMEMORY = (-5)
SP_PARITY = &H1&
SP_PARITY_CHECK = &H20&
SP_RLSD = &H40&
SP_SERIALCOMM = &H1&
SP_STOPBITS = &H8&
SP_USERABORT = (-3)
SPACEPARITY = 4
STARTF_FORCEOFFFEEDBACK = &H80
STARTF_FORCEONFEEDBACK = &H40
STARTF_RUNFULLSCREEN = &H20
STARTF_USECOUNTCHARS = &H8
STARTF_USEFILLATTRIBUTE = &H10
STARTF_USEPOSITION = &H4
STARTF_USESHOWWINDOW = &H1
STARTF_USESIZE = &H2
STARTF_USESTDHANDLES = &H100
STD_ERROR_HANDLE = -12&
STD_INPUT_HANDLE = -10&
STD_OUTPUT_HANDLE = -11&
STOPBITS_10 = &H1&
STOPBITS_15 = &H2&
STOPBITS_20 = &H4&
STREAM_CONTAINS_SECURITY = &H2
STREAM_MODIFIED_WHEN_READ = &H1
STRETCH_ANDSCANS = 1
TC_CP_STROKE = &H4
TC_CR_90 = &H8
TC_CR_ANY = &H10
TC_EA_DOUBLE = &H200
TC_GP_TRAP = 2
TC_HARDERR = 1
TC_IA_ABLE = &H400
TC_NORMAL = 0
TC_OP_CHARACTER = &H1
TC_OP_STROKE = &H2
TC_RA_ABLE = &H2000
TC_RESERVED = &H8000
TC_SA_CONTIN = &H100
TC_SA_DOUBLE = &H40
TC_SA_INTEGER = &H80
TC_SCROLLBLT = &H10000
TC_SF_X_YINDEP = &H20
TC_SIGNAL = 3
TC_SO_ABLE = &H1000
TC_UA_ABLE = &H800
TC_VA_ABLE = &H4000
TCI_SRCCHARSET = 1
TCI_SRCCODEPAGE = 2
TCI_SRCFONTSIG = 3
THREAD_BASE_PRIORITY_IDLE = -15
THREAD_BASE_PRIORITY_LOWRT = 15
THREAD_BASE_PRIORITY_MAX = 2
THREAD_BASE_PRIORITY_MIN = -2
THREAD_PRIORITY_ABOVE_NORMAL = (THREAD_PRIORITY_HIGHEST - 1)
THREAD_PRIORITY_BELOW_NORMAL = (THREAD_PRIORITY_LOWEST + 1)
THREAD_PRIORITY_ERROR_RETURN = (MAXLONG)
THREAD_PRIORITY_HIGHEST = THREAD_BASE_PRIORITY_MAX
THREAD_PRIORITY_IDLE = THREAD_BASE_PRIORITY_IDLE
THREAD_PRIORITY_LOWEST = THREAD_BASE_PRIORITY_MIN
THREAD_PRIORITY_NORMAL = 0
THREAD_PRIORITY_TIME_CRITICAL = THREAD_BASE_PRIORITY_LOWRT

TIME_BYTES = &H4
TIME_FORCE24HOURFORMAT = &H8
TIME_MIDI = &H10
TIME_MS = &H1
TIME_NOMINUTESORSECONDS = &H1
TIME_NOSECONDS = &H2
TIME_NOTIMEMARKER = &H4
TIME_ONESHOT = 0
TIME_PERIODIC = 1
TIME_SAMPLES = &H2
TIME_SMPTE = &H8
TIMEOUT_ASYNC = &HFFFF
TIMERR_BASE = 96
TIMERR_NOCANDO = (TIMERR_BASE + 1)
TIMERR_NOERROR = (0)
TIMERR_STRUCT = (TIMERR_BASE + 33)
TLS_OUT_OF_INDEXES = &HFFFF
TRUNCATE_EXISTING = 5
TWOSTOPBITS = 2
UNLOAD_DLL_DEBUG_EVENT = 7
VER_PLATFORM_WIN32_NT = 2
VER_PLATFORM_WIN32_WINDOWS = 1
VER_PLATFORM_WIN32s = 0
VERTRES = 10
VERTSIZE = 6

●Win32メッセージ定数一覧

WM_ACTIVATE = &H6
WM_ACTIVATEAPP = &H1C
WM_ASKCBFORMATNAME = &H30C
WM_CANCELJOURNAL = &H4B
WM_CANCELMODE = &H1F
WM_CHANGECBCHAIN = &H30D
WM_CHAR = &H102
WM_CHARTOITEM = &H2F
WM_CHILDACTIVATE = &H22
WM_CHOOSEFONT_GETLOGFONT = (WM_USER + 1)
WM_CHOOSEFONT_SETFLAGS = (WM_USER + 102)
WM_CHOOSEFONT_SETLOGFONT = (WM_USER + 101)
WM_CLEAR = &H303
WM_CLOSE = &H10
WM_COMMAND = &H111
WM_COMMNOTIFY = &H44
WM_COMPACTING = &H41
WM_COMPAREITEM = &H39
WM_CONVERTREQUESTEX = &H108
WM_COPY = &H301
WM_COPYDATA = &H4A
WM_CREATE = &H1
WM_CTLCOLORBTN = &H135
WM_CTLCOLORDLG = &H136
WM_CTLCOLOREDIT = &H133
WM_CTLCOLORLISTBOX = &H134
WM_CTLCOLORMSGBOX = &H132
WM_CTLCOLORSCROLLBAR = &H137
WM_CTLCOLORSTATIC = &H138
WM_CUT = &H300
WM_DDE_ACK = (WM_DDE_FIRST + 4)
WM_DDE_ADVISE = (WM_DDE_FIRST + 2)
WM_DDE_DATA = (WM_DDE_FIRST + 5)
WM_DDE_EXECUTE = (WM_DDE_FIRST + 8)
WM_DDE_FIRST = &H3E0
WM_DDE_INITIATE = (WM_DDE_FIRST)
WM_DDE_LAST = (WM_DDE_FIRST + 8)
WM_DDE_POKE = (WM_DDE_FIRST + 7)
WM_DDE_REQUEST = (WM_DDE_FIRST + 6)
WM_DDE_TERMINATE = (WM_DDE_FIRST + 1)
WM_DDE_UNADVISE = (WM_DDE_FIRST + 3)
WM_DEADCHAR = &H103
WM_DELETEITEM = &H2D
WM_DESTROY = &H2
WM_DESTROYCLIPBOARD = &H307
WM_DEVMODECHANGE = &H1B
WM_DRAWCLIPBOARD = &H308
WM_DRAWITEM = &H2B
WM_DROPFILES = &H233
WM_ENABLE = &HA
WM_ENDSESSION = &H16
WM_ENTERIDLE = &H121
WM_ENTERMENULOOP = &H211
WM_ERASEBKGND = &H14
WM_EXITMENULOOP = &H212
WM_FONTCHANGE = &H1D
WM_GETDLGCODE = &H87
WM_GETFONT = &H31
WM_GETHOTKEY = &H33

WM_GETMINMAXINFO = &H24
WM_GETTEXT = &HD
WM_GETTEXTLENGTH = &HE
WM_HOTKEY = &H312
WM_HSCROLL = &H114
WM_HSCROLLCLIPBOARD = &H30E
WM_ICONERASEBKGND = &H27
WM_IME_CHAR = &H286
WM_IME_COMPOSITION = &H10F
WM_IME_COMPOSITIONFULL = &H284
WM_IME_CONTROL = &H283
WM_IME_ENDCOMPOSITION = &H10E
WM_IME_KEYDOWN = &H290
WM_IME_KEYLAST = &H10F
WM_IME_KEYUP = &H291
WM_IME_NOTIFY = &H282
WM_IME_SELECT = &H285
WM_IME_SETCONTEXT = &H281
WM_IME_STARTCOMPOSITION = &H10D
WM_INITDIALOG = &H110
WM_INITMENU = &H116
WM_INITMENUPOPUP = &H117
WM_KEYDOWN = &H100
WM_KEYFIRST = &H100
WM_KEYLAST = &H108
WM_KEYUP = &H101
WM_KILLFOCUS = &H8
WM_LBUTTONDBLCLK = &H203
WM_LBUTTONDOWN = &H201
WM_LBUTTONUP = &H202
WM_MBUTTONDBLCLK = &H209
WM_MBUTTONDOWN = &H207
WM_MBUTTONUP = &H208
WM_MDIACTIVATE = &H222
WM_MDICASCADE = &H227
WM_MDICREATE = &H220
WM_MDIDESTROY = &H221
WM_MDIGETACTIVE = &H229
WM_MDIICONARRANGE = &H228
WM_MDIMAXIMIZE = &H225
WM_MDINEXT = &H224
WM_MDIREFRESHMENU = &H234
WM_MDIRESTORE = &H223
WM_MDISETMENU = &H230
WM_MDITILE = &H226
WM_MEASUREITEM = &H2C
WM_MENUCHAR = &H120
WM_MENUSELECT = &H11F
WM_MOUSEACTIVATE = &H21
WM_MOUSEFIRST = &H200
WM_MOUSELAST = &H209
WM_MOUSEMOVE = &H200
WM_MOVE = &H3
WM_NCACTIVATE = &H86
WM_NCCALCSIZE = &H83
WM_NCCREATE = &H81
WM_NCDESTROY = &H82
WM_NCHITTEST = &H84
WM_NCLBUTTONDBLCLK = &HA3
WM_NCLBUTTONDOWN = &HA1
WM_NCLBUTTONUP = &HA2
WM_NCMBUTTONDBLCLK = &HA9
WM_NCMBUTTONDOWN = &HA7
WM_NCMBUTTONUP = &HA8
WM_NCMOUSEMOVE = &HA0
WM_NCPAINT = &H85
WM_NCRBUTTONDBLCLK = &HA6
WM_NCRBUTTONDOWN = &HA4
WM_NCRBUTTONUP = &HA5
WM_NEXTDLGCTL = &H28
WM_NULL = &H0
WM_OTHERWINDOWCREATED = &H42
WM_OTHERWINDOWDESTROYED = &H43
WM_PAINT = &HF
WM_PAINTCLIPBOARD = &H309
WM_PAINTICON = &H26
WM_PALETTECHANGED = &H311
WM_PALETTEISCHANGING = &H310
WM_PARENTNOTIFY = &H210
WM_PASTE = &H302
WM_PENWINFIRST = &H380
WM_PENWINLAST = &H38F
WM_POWER = &H48
WM_PSD_ENVSTAMPRECT = (WM_USER + 5)
WM_PSD_FULLPAGERECT = (WM_USER + 1)
WM_PSD_GREEKTEXTRECT = (WM_USER + 4)
WM_PSD_MARGINRECT = (WM_USER + 3)
WM_PSD_MINMARGINRECT = (WM_USER + 2)
WM_PSD_PAGESETUPDLG = (WM_USER)
WM_PSD_YAFULLPAGERECT = (WM_USER + 6)

WM_QUERYDRAGICON = &H37
WM_QUERYENDSESSION = &H11
WM_QUERYNEWPALETTE = &H30F
WM_QUERYOPEN = &H13
WM_QUEUESYNC = &H23
WM_QUIT = &H12
WM_RBUTTONDBLCLK = &H206
WM_RBUTTONDOWN = &H204
WM_RBUTTONUP = &H205
WM_RENDERALLFORMATS = &H306
WM_RENDERFORMAT = &H305
WM_SETCURSOR = &H20
WM_SETFOCUS = &H7
WM_SETFONT = &H30
WM_SETHOTKEY = &H32
WM_SETREDRAW = &HB
WM_SETTEXT = &HC
WM_SHOWWINDOW = &H18
WM_SIZE = &H5
WM_SIZECLIPBOARD = &H30B
WM_SPOOLERSTATUS = &H2A
WM_SYSCHAR = &H106
WM_SYSCOLORCHANGE = &H15
WM_SYSCOMMAND = &H112
WM_SYSDEADCHAR = &H107
WM_SYSKEYDOWN = &H104
WM_SYSKEYUP = &H105
WM_TIMECHANGE = &H1E
WM_TIMER = &H113
WM_UNDO = &H304
WM_USER = &H400
WM_VKEYTOITEM = &H2E
WM_VSCROLL = &H115
WM_VSCROLLCLIPBOARD = &H30A
WM_WINDOWPOSCHANGED = &H47
WM_WINDOWPOSCHANGING = &H46
WM_WININICHANGE = &H1A

索　引

英記号

あ

か

さ

た

な

は

ま

や・ら・わ

Microsoft Visual C++／Visual Basic
Win32 API オフィシャルリファレンス

1996年6月11日　初版発行

監　修　マイクロソフト株式会社
編　者　アスキー書籍編集部

発行所　**株式会社アスキー**
〒151-24　東京都渋谷区代々木4-33-10

Printed in Japan

254-400-905